ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA INFORMÁTICA

IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN HIDROLÓGICA EN UN ROBOT SUBACUÁTICO

Alejandro Chacón Peregrino

2 de febrero de 2017

ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA EN INFORMÁTICA

IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN HIDROLÓGICA EN UN ROBOT SUBACUÁTICO

- Departamento: Ingeniería Informática.
- Director del proyecto: Andrés Yañez Escolano
- Codirector del proyecto: Santiago García López
- Autor del proyecto: Alejandro Chacón Peregrino

Cádiz, 2 de febrero de 2017

Fdo: Alejandro Chacón Peregrino

Índice general

Índice de figuras

Índice de cuadros

IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN HIDROLÓGICA EN UN ROBOT SUBACUÁTICO

REF: 000001

MEMORIA

- **CLIENTE:** UNIVERSIDAD DE CÁDIZ (ESCUELA SUPERIOR DE INGENIERÍA)
  AVENIDA DE LA UNIVERSIDAD DE CÁDIZ Nº 10, 11519 PUERTO REAL
  956 48 32 00
  DIRECCION.ESI@UCA.ES

- **AUTOR:** ALEJANDRO CHACON PEREGRINO
  INGENIERO INFORMÁTICO
  DNI 32079315L
  ALEJANDRO.CHACONPEREGRINO@ALUM.UCA.ES

Cádiz, 2 de febrero de 2017

Índice

1. Introducción

La hidrología es la ciencia que estudia las aguas de la Tierra, su ocurrencia, circulación y distribución, sus propiedades físicas y químicas y su influencia sobre el medio ambiente, incluyendo su relación con los seres vivientes. El dominio de la hidrología abarca la historia completa de la existencia del agua sobre la Tierra (U.S. Federal Council for Science and Tecnology, 1962).

Uno de los principales objetivos de la hidrología es la monitorización con el objetivo de identificar el origen del agua, el desplazamiento de masas de agua, los fenómenos de mezcla y los procesos de tipo físico-químico y biológico que pueden afectar a la calidad del recurso. Para ello, se requiere la realización de mediciones de diversas variables como son el pH, la conductividad eléctrica, la temperatura y el potencial de óxido-reducción en los cuerpos de agua tanto naturales como artificiales.

La adquisición de datos in situ no suele ser una tarea fácil debido a las dificultades derivadas de la extensión y accesibilidad del medio. Las medidas de los parámetros físico-químicos en las masas de agua superficiales se suele realizar mediante sondas de medida directa o dispositivos de registro automático que requieren de su colocación en las ubicaciones seleccionadas por procedimientos manuales, desde la orilla o desde embarcaciones, puentes u otras estructuras. Esto evidentemente limita y dificulta la obtención de los datos.

En ambos casos, el disponer de una plataforma móvil, controlable de forma remota, que disponga de sondas para el registro automatizado de las variables señaladas a lo largo de perfiles, con registro continuo de la profundidad y orientación de la trayectoria, puede suponer un avance significativo en las labores de adquisición de datos.

2. Objetivo

El principal objetivo de este proyecto es el diseño e implementación de un sistema de monitorización hidrológica en un ROV (acrónimo del inglés Remoted Operated Vehicle) de bajo coste para realizar tareas de adquisición de datos relativos a las variables físico-químicas que habitualmente se consideran en hidrología como básicas o fundamentales (temperatura, conductividad eléctrica, pH y potencial redox).

Este sistema podrá ser manejado de forma remota a través de la propia interfaz del ROV y permitirá el muestreo de las variables citadas. Los valores obtenidos en el proceso de medida serán mostrados al usuario a través de la interfaz en tiempo real, tanto de forma numérica como de forma gráfica. La información se visualizará en pantalla de forma simultánea a la imagen registrada por la cámara de alta resolución a bordo del vehículo, lo que permitirá operarlo para adaptar el movimiento a las necesidades de la adquisición.

Además los datos adquiridos serán almacenados en la memoria sólida del dispositivo para su posterior descarga y explotación. Por otra parte, se plantea dotar al sistema de elementos para la mejora del posicionamiento (orientación y GPS) y un dispositivo de iluminación complementario y más potente que el que dispone el ROV original, que sea manejable desde la interfaz del usuario.

3. Antecedentes

Habitualmente las medidas de variables hidrológicas se llevan a cabo bien con dispositivos de utilización manual (termómetros, conductivímetros, pHmetros y sondas multiparamétricas) que sirven para realizar medidas puntuales en la superficie o en la columna de agua (según la longitud del cable utilizado), bien con dispositivos de registro continuo ubicados en posiciones fijas, suspendidos de boyas o de otros elementos.

Para la obtención de información hidrológica distribuida espacialmente dentro de los cuerpos de agua se proyectan perfiles longitudinales que requieren la planificación y realización de tareas rutinarias y laboriosas con la asistencia de embarcaciones y de personal. La toma de datos se planifica de forma sistemática con el inconveniente de que resulta difícil alterar el patrón de muestreo durante la adquisición, según los resultados que se van obteniendo.

La utilización de vehículos operados de forma remota puede simplificar las labores de toma de datos en el interior de un cuerpo de agua, aportando una mayor rapidez, flexibilidad y adaptabilidad al proceso de adquisición. La obtención de los registros de las variables mientras se visualiza en la interfaz su evolución espacial en combinación con la imagen capturada por la cámara de alta definición posibilita que el usuario opere en tiempo real el vehículo que porta los sensores y pueda aprovechar su capacidad de desplazamiento tridimensional.

4. Descripción de la Situación Actual

La relativamente reciente y creciente participación de drones y ROVs en actividades empresariales, de investigación y lúdicas ha facilitado el acceso a zonas que presentaban gran dificultad o eran imposibles para el ser humano. Esto vehículos pueden ser equipados con sensores que les permiten muestrear extensas zonas de forma rápida, económica y eficiente.

En en los siguientes apartados se realiza una breve descripción de los vehículos subacuáticos que pueden ser empleados en estas tareas y de los parámetros que habitualmente son susceptibles de ser monitorizados.

4.1. Vehículos de control remoto subacuáticos

Estos vehículos son controlados de manera remota a través de un cable que conecta el ROV y el puesto de control. Suelen llevar una cámara incorporada para transmitir en tiempo real y ayudar al manejo y la observación.

Los ROVs se pueden clasificar en dos clases principalmente:

4.1.1. Vehículo de trabajo (Work Class)

Estos vehículos son de medio y gran tamaño. Están diseñados para sumergirse en grandes profundidades (actualmente hay algunos que soportan hasta los 7000 metros).

Dependiendo de la actividad a realizar, disponen de varios brazos mecánicos y herramientas que permiten realizar diferentes trabajos, como atornillar y taladrar.

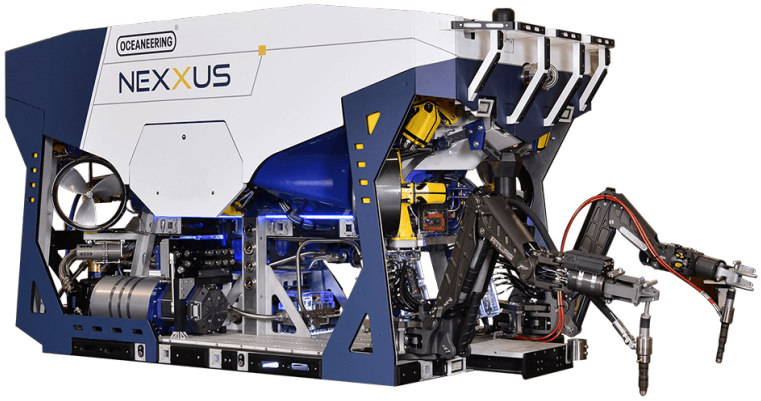


Figura 4.1: ROV de trabajo.

La utilización de estos vehículos está muy extendida en determinados ámbitos tecnológicos: exploración y producción petrolífera, tendido de cables submarinos, recuperación de pecios y restos de accidentes marítimos y aéreos, etc.

Habitualmente se trata de equipos sofisticados de elevado coste (llegando a costar millones de euros), que requieren de infraestructura de apoyo relativamente compleja. Todo ello limita notablemente su disponibilidad y uso, haciendo que su empleo quede restringido a campos de elevado interés económico.

4.1.2. Vehículo de observación (Observation Class)

Estos vehículos son de pequeño tamaño ya que están diseñados para poder entrar en zonas de difícil acceso para su observación o estudio.

Incorporan una o más cámaras que envían simultáneamente y algunos modelos disponen de un pequeño brazo para tomar muestras. Como máximo, soportan hasta 300 metros de profundidad.

Figura 4.2: ROV de observación.

Debido a su pequeño tamaño y peso, pueden disponer de varias baterías incorporadas con autonomía de varias horas. Su precio varía desde menos de mil euros (pequeños ROVs de bajo coste) hasta varios miles de euros.

A continuación se muestran unos ejemplos de vehículos de bajo coste que podrían ser interesantes para el desarrollo de este proyecto.

4.1.2.1. BlueROV

Este vehículo de bajo coste dispone de seis motores que lo dotan de un manejo muy preciso y potente. Tanto su software como su hardware son de código abierto, por lo que permite su total modificación. Tiene un chasis parecido a los vehículos profesionales aunque está diseñado para ser ampliable por el usuario.

Su sistema eléctrico se encuentra en el interior de un tubo central atornillado para protegerla del agua. Su controlador es una Raspberry Pi que funciona bajo sistema Linux. Dispone de un cable de 300 metros, aunque la profundidad máxima para garantizar su funcionamiento son 100 metros.

Figura 4.3: BlueROV.

Dispone de una cámara HD 1080p que retransmite en directo a baja latencia a la interfaz de control. Su interfaz es multiplataforma por lo que puede ser usado desde cualquier sistema operativo. Usa baterías de larga duración facilmente intercambiables.

Su precio es 4.500€ totalmente montado y listo para sumergir [30].

4.1.2.2. OpenROV

Al contrario que la opción anterior solo dispone de tres motores por lo que no tiene un manejo tan preciso. Por el contrario, su chasis es más básico por lo que permite añadir elementos de forma más sencilla por el usuario. Su software y hardware son también de código abierto por lo que es totalmente modificable.

Igual que el BlueROV, su sistema eléctrico y de control se encuentra dentro de un tubo central. El controlador en este caso es una Beaglebone Black que funciona bajo sistema Linux. Además incorpora una tarjeta Arduino MEGA para controlar los motores y poder conectar más elementos. Este tubo está sellado por juntas tóricas y dispone de un pequeño orificio por el que se puede extraer aire para dejarlo al vacío.

Dispone también de 300 metros de cable para sumergirse hasta 100 metros de profundidad, que son los recomendados por el fabricante.

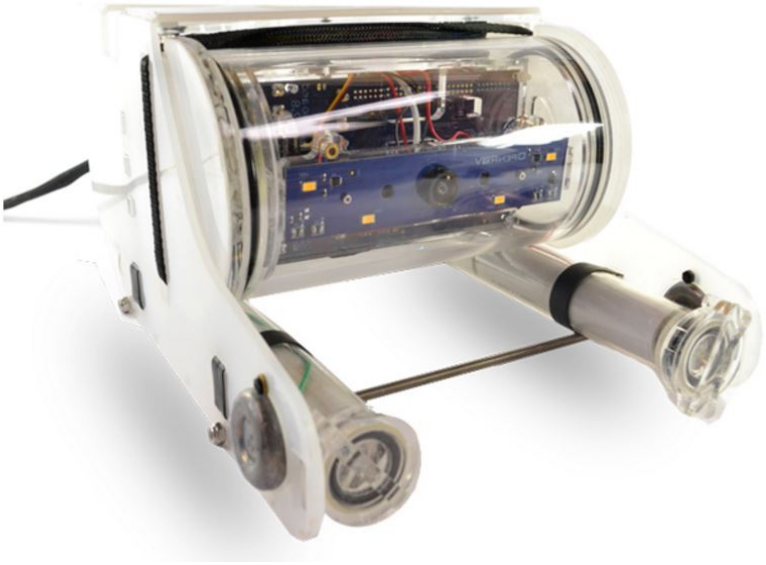


Figura 4.4: OpenROV.

Su interfaz de control se utiliza mediante un navegador web, preferiblemente Google Chrome, por lo que puede ser usada en cualquier plataforma que admita dicho navegador, incluso en sistemas Android. Tiene una cámara HD que retransmite en directo a dicha interfaz y además dispone de cuatro LEDs para mejorar la visión en zonas de baja iluminación. Su batería dura de dos a tres horas, dependiendo de la actividad y es fácilmente intercambiable.

Este modelo puede adquirirse de dos formas. Totalmente montado y funcional por un precio de 1.451 euros y como un kit de montaje por un precio de 900 euros [31].

Por su bajo coste y las prestaciones ofrecidas, se ha optado por elegir este ROV para el desarrollo del proyecto.

4.2. Sensores de medición hidrológica

Las propiedades del agua que habitualmente se miden un estudio hidrológico son temperatura, conductividad eléctrica, pH y potencial de óxido-reducción. Permiten una rápida caracterización del cuerpo de agua, de su naturaleza y de los procesos que en su seno tienen lugar, y resultan imprescindibles cuando se pretende modelizar los procesos físicos, geoquímicos y biológicos que tienen lugar en el medio.

4.2.1. Sensor de pH

Este sensor se utiliza para medir el pH en una disolución. El pH se define como el logaritmo de la actividad de hidrogeniones con signo negativo. El pH varía aproximadamente el 8 % con el aumento de cada grado de temperatura, por lo que hay que referirlo a una temperatura determinada, normalmente 25°C. El pH del agua pura a 25°C es 7,00.

La medida se realiza electrométricamente, mediante la diferencia de potencial eléctrico entre dos electrodos. El dispositivo consta de un electrodo de calomel, llamado electrodo de referencia, que tiene un potencial de equilibrio estable y conocido y un electrodo de vidrio polarizable sensible al ion de hidrógeno.

4.2.2. Sensor de conductividad eléctrica (EC)

Este sensor se utiliza para medir la capacidad de circulación de la corriente eléctrica que tiene una disolución. Esta propiedad se mide de forma estandarizada como la conductividad que existe entre dos electrodos paralelos de 1 cm^2 de superficie cada uno, separados 1 cm, dispuestos en el seno de la disolución a medir de forma que el medio se pueda considerar infinito.

El agua pura es muy poco conductora, mientras que pequeñas concentraciones de sales disueltas aumentan significativamente esta propiedad. La utilidad de medir esta propiedad en campo estriba en que es una forma rápida y sencilla de conocer la concentración de sales en el agua. Además de la salinidad, en la conductividad influye la temperatura.

Para realizar la medida se utilizan dos electrodos de platino que miden la resistencia eléctrica de una disolución al ser expuesta a corriente alterna. Donde R es la resistencia eléctrica medida por los electrodos, l es la distancia entre los electrodos y A es su área.

$$\kappa = \frac{1}{R} \cdot \frac{l}{A}$$

Dado que la conductividad eléctrica se ve afectada por la temperatura a la que está una disolución se debe emplear un sensor de temperatura a la vez. La conductividad final se obtiene de la siguiente ecuación.

$$\sigma_T = \sigma_{T_{cal}}[1 + \alpha(T - T_{cal})]$$

Donde T es la temperatura de la disolución, T_{cal} es la temperatura de la disolución de calibración, $\sigma_{T_{cal}}$ es la conductividad eléctrica en la temperatura de la disolución de calibración y α es la pendiente de compensación de la disolución.

4.2.3. Sensor de potencial de óxido-reducción (ORP)

El potencial redox de un sistema mide la estabilidad de un ion en un nivel de oxidación determinado. En los procesos de disolución de sales por el ataque del agua a los minerales, son de gran interés aquellos mecanismos en los que intervienen sustancias que cambian o pueden cambiar su estado de valencia, oxidándose o reduciéndose.

La medida se realiza de forma similar al pH, por métodos electrométricos. El sensor dispone de dos electrodos, un electrodo de referencia y otro de un metal noble como oro, plata o platino.

5. Normas y Referencias

5.1. Disposiciones legales y Normas aplicadas

- **UNE 157801:2007** - Criterios generales para la elaboración de proyectos de sistemas de información.
- **UNE 157001:2014** - Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico.

5.2. Bibliografía

[1] **Adafruit**. HMC5883L Triple-Axis Magnetometer Compass Sensor. https://learn.adafruit.com/adafruit-hmc5883l-breakout-triple-axis-magnetometer-compass-sensor/overview. [02-02-2017].

[2] **Amazon**. Cinta eléctrica autosoldable de goma. https://www.amazon.es/gp/product/B0014DVH5Y/ref=oh_aui_detailpage_o03_s00?ie=UTF8&psc=1. [02-02-2017].

[3] **Amazon**. Floureon 7.4V Batería. https://www.amazon.es/Floureon-Bater%C3%ADa-Hobbies-Compatible-Helic%C3%B3ptero/dp/B013B2DOYW. [02-02-2017].

[4] **Arduino**. Arduino - Millis. https://www.arduino.cc/en/Reference/Millis. [02-02-2017].

[5] **Arduino**. Arduino DUE Specifications. https://www.arduino.cc/en/Main/arduinoBoardDue. [02-02-2017].

[6] **Arduino**. Arduino MEGA Specifications. https://www.arduino.cc/en/Main/arduinoBoardMega. [02-02-2017].

[7] **Arduino**. Arduino NANO Specifications. https://www.arduino.cc/en/Main/ArduinoBoardNano. [02-02-2017].

[8] **Arduino**. Arduino UNO Specifications. https://www.arduino.cc/en/Main/arduinoBoardUno. [02-02-2017].

[9] **Arduino**. EEPROM Library. https://www.arduino.cc/en/Reference/EEPROM. [02-02-2017].

[10] **Arduino**. SD Library. https://www.arduino.cc/en/Reference/SD. [02-02-2017].

[11] **Arduino**. SoftwareSerial Library. https://www.arduino.cc/en/Reference/SoftwareSerial. [02-02-2017].

[12] **Arduino**. Wire Library. https://www.arduino.cc/en/Reference/Wire. [02-02-2017].

[13] **DFRobots**. Sensor EC. https://www.dfrobot.com/index.php?route=product/product&product_id=1123. [02-02-2017].

[14] **DFRobots**. Sensor ORP. https://www.dfrobot.com/index.php?route=product/product&product_id=1071. [02-02-2017].

[15] **DFRobots**. Sensor pH. https://www.dfrobot.com/index.php?route=product/product&product_id=1110. [02-02-2017].

[16] **Ebay**. GY-NEO6MV2 NEO-6M Módulo GPS. http://www.ebay.es/itm/GY-NEO6MV2-NEO-6M-GPS-Modulo-NEO6MV2-Vuelo-Controlador-EEPROM-con-USB2TTL-Regalo-/322135963604. [02-02-2017].

[17] **Ebay**. Kit 6x Cable 1m AWG24. http://www.ebay.es/itm/6-Cables-de-1-Metro-AWG24-WRAPPING-WIRE-Arduino-Electronic-Puente-Robotica-CA16-/351774946480?_trksid=p2141725.m3641.l6368. [02-02-2017].

[18] **Ebay**. MicroSD Arduino. http://www.ebay.es/itm/Micro-SD-Arduino-MODULO-LECTOR-GRABADOR-TARJETA-MicroSD-CARD-ARDUINO-read-write-/272539521045. [02-02-2017].

[19] **Ebay**. Modulo arduino brújula GY-273 HMC5883L. http://www.ebay.es/itm/Modulo-arduino-brujula-GY-273-HMC5883L-Triple-Axis-Compas-Magnetometer-/252332670437. [02-02-2017].

[20] **Electrodos y sensores**. Sensors Documentation. http://www.electrodosysensores.com/. [02-02-2017].

[21] **Farnell**. Transistor NPN 2N2222. http://es.farnell.com/multicomp/2n2222/transistor-npn-30v-800ma-to-18/dp/9206884. [02-02-2017].

[22] **GitHub**. A simple theme for OpenROV. https://github.com/joakar/openrov-simple-theme. [02-02-2017].

[23] **GitHub**. HMC5883L Simple Library. https://github.com/sleemanj/HMC5883L_Simple. [02-02-2017].

[24] **GitHub**. OpenROV. https://github.com/OpenROV. [02-02-2017].

[25] **GitHub**. TinyGPS++. https://github.com/mikalhart/TinyGPSPlus. [02-02-2017].

[26] **Leroy Merlin**. Balda rectangular metacrilato 60X20cm. http://www.leroymerlin.es/fp/15215900/. [02-02-2017].

[27] **Leroy Merlin**. Rosca 50mm. http://www.leroymerlin.es/fp/12154506/. [02-02-2017].

[28] **Leroy Merlin**. Tubo rígido PVC evacuación 50 mm y 2 m. http://www.leroymerlin.es/fp/14622132/. [02-02-2017].

[29] **Naylamp Mechatronics**. Arduino y memoria SD. http://www.naylampmechatronics.com/blog/38_Tutorial-Arduino-y-memoria-SD-y-micro-SD-.html. [02-02-2017].

[30] **NidoRobotics**. BlueROV. https://nidorobotics.com/producto/bluerov2/. [02-02-2017].

[31] **NidoRobotics**. OpenROV. https://nidorobotics.com/producto/openrov-v2-8-kit/. [02-02-2017].

[32] **Node.js**. Node.js Documentation. https://nodejs.org/docs/latest/api/documentation.html. [02-02-2017].

[33] **OpenROV**. How to create a software plugin for OpenROV Cockpit. http://openrov.dozuki.com/Guide/How+to+create+a+software+plugin+for+OpenROV+Cockpit/22. [02-02-2017].

[34] **Robert D., Christ y Robert L., Wernli Sr**. The ROV Manual. *Elsevier*, 2007.

[35] **RS Components**. Arduino DUE. http://es.rs-online.com/web/p/kits-de-desarrollo-de-procesador-y-microcontrolador/7697412/. [02-02-2017].

[36] **RS Components**. Arduino MEGA. http://es.rs-online.com/web/p/kits-de-desarrollo-de-procesador-y-microcontrolador/7154084/. [02-02-2017].

[37] **RS Components**. Arduino NANO. http://es.rs-online.com/web/p/kits-de-desarrollo-de-procesador-y-microcontrolador/6961667/. [02-02-2017].

[38] **RS Components**. Arduino UNO. http://es.rs-online.com/web/p/kits-de-desarrollo-de-procesador-y-microcontrolador/7154081/. [02-02-2017].

[39] **RS Components**. LED Blanco 5mm. http://es.rs-online.com/web/p/led-visibles/7133955/. [02-02-2017].

[40] **RS Components**. Resistencia 100Ω, 0,25W. http://es.rs-online.com/web/p/resistencias-fijas-de-orificio-pasante/0135774/. [02-02-2017].

[41] **RS Components**. Resistencia 33Ω, 0,25W. http://es.rs-online.com/web/p/resistencias-fijas-de-orificio-pasante/8066499/. [02-02-2017].

[42] **SparkFun**. I2C. https://learn.sparkfun.com/tutorials/i2c. [02-02-2017].

[43] **SparkFun**. SPI. https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi. [02-02-2017].

[44] **WebSockets**. Introducción a WebSockets. hhttps://sites.google.com/site/gabineteutn/investigacion-y-desarrollo/html5/tutoriales/introduccion-a-websocket. [02-02-2017].

5.3. Herramientas

5.3.1. Herramientas software

Arduino IDE - Entorno de desarrollo para programar las microcontroladoras Arduino.

C - Lenguaje de programación de alto nivel. Necesario para programar las microcontroladoras usadas en el proyecto.

Dia - Software de diseño de diagramas UML.

Eagle - Software para creación de circuitos electrónicos. Utilizado para el diseño de las placas PCB.

Fritzing - Software de diseño electrónico que ofrece la posibilidad de crear diseños finales a partir del esquemático. Usado para la creación de diagramas.

GanttProject - Software de diseño de diagramas de Gantt.

JavaScript - Lenguaje de programación interpretado. Usado en la interfaz del OpenROV.

Microsoft Visio - Software de dibujo vectorial usado para el diseño de dibujos y diagramas.

Sublime Text - Editor de texto especializado en código de programación.

WinSCP - Cliente SFTP para realizar conexiones a través de FTP y SSH.

5.3.2. Herramientas mecánicas

Fresadora PCB LPKF ProtoMat S103 - Fresadora mecánica utilizada para obtener las placas PCB.

5.4. Otras Referencias

5.4.1. Repositorio en GitHub

Toda la documentación y software se encuentran alojados en GitHub, en el siguiente enlace: https://github.com/alhenx/tfg

6. Definiciones y Abreviaturas

Arduino Microcontroladora de hardware libre de fácil uso.

CSS Cascading Stylesheets (Hoja de estilos en cascada).

EC Electrical Conductivity (Conductividad eléctrica). Medida de la capacidad de un material o sustancia para dejar pasar libremente la corriente eléctrica.

Ethernet Estándar de red local para computadores (IEEE 802.3).

FTP File Transfer Protocol (Protocolo de transferencia de archivos).

GND Ground (Tierra).

GPS Global Positioning System (Sistema de posicionamiento global). Sistema que permite calcular la posición de un objeto en la Tierra.

HD High Definition (Alta definición).

I2C Inter-Integrated Circuit (Circuito inter-integrado).

LED Light-Emitting Diode (Diodo emisor de luz).

LIPO Lithium Polymer Battery (Batería de polímero de litio).

MISO Master Input Slave Output (Entrada de maestro y salida de esclavo).

MicroSD Tarjeta de memoria flash de pequeño tamaño.

MOSI Master Output Slave Input (Salida de maestro y entrada de esclavo).

ORP Oxidation/Reduction Potential (Potencial de reducción/oxidación). Medida de la capacidad de oxidación y reducción de una solución acuosa.

PCB Printed Circuit Board (Placa de circuito impreso).

pH Medida de acidez o alcalinidad de una disolución.

ROV Remote Operated Vehicle (Vehículo operado a distancia).

RX Reception (Recepción).

SDA Serial Data (Datos serial).

SCL, SCLK, SCK Serial Clock (Reloj serial).

SPI Serial Peripheral Interface (Interfaz serial periférica).

SS, CS Slave Select (Selección de esclavo).

SSH Secure Shell (Intérprete de órdenes seguro).

TX Transmission (Transmisión).

VIN Voltage In (Entrada de voltaje).

7. Requisitos Iniciales

Los requisitos iniciales requeridos para el proyecto por el cliente son los siguientes:

- Monitorización de variables hidrológicas.
- Capacidad para almacenar los datos.
- Posicionamiento del ROV.
- Capacidad de iluminación externa.
- Módulo externo de fácil instalación.

Estos requisitos son ampliamente detallados en el apartado 19, “Especificaciones del Sistema”.

8. Alcance

El alcance del proyecto engloba los siguientes puntos:

- Montaje del robot subacuático.
- Análisis del software de control del robot subacuático.
- Desarrollo de la especificación de requisitos del sistema y su análisis.
- Modificación del software de control del robot para la integración de la plataforma de monitorización.
- Modificación del software de control del robot para automatizar el proceso de desplazamiento (línea recta, giros...)
- Análisis, diseño e implementación de la comunicación entre el módulo externo y el ROV.
- Diseño e implementación de la plataforma de monitorización.

9. Estudio de Alternativas y Viabilidad

Durante el desarrollo del proyecto se han estudiado diferentes alternativas de cara al diseño e implementación de los diferentes componentes. A continuación se describen.

9.1. Plataforma del módulo y conexión

Durante el proceso de ensamblaje del robot quedaron sin uso seis cables auxiliares para conectar elementos extras, por lo que la primera opción evaluada fue conectar directamente los sensores al módulo central.

Esta opción fue descartada. Seis cables eran insuficientes para añadir todos los sensores necesarios, por lo que se pensó en hacer uso de un dispositivo que se conectara al nodo central y enviara los datos.

Este dispositivo debía cumplir los siguientes requisitos:

- Pequeño tamaño.
- Bajo coste.
- Permitir la conexión a la tarjeta controladora del ROV.
- Realizar el envío de datos de forma bidireccional.
- Conectar los diferentes sensores.

Se optó por usar otra controladora Arduino para asegurar la compatibilidad. Entre las diferentes posibilidades se analizaron las siguientes:

Modelo	Tamaño (mm)	Precio	Pines
Arduino UNO [8]	68,6 x 53,4	14 digitales y 6 analógicos	21,25 € [38]
Arduino NANO [7]	45 x 18	14 digitales y 8 analógicos	18,36 € [37]
Arduino MEGA [6]	101,5 x 53,3	54 digitales y 16 analógicos	43,51 € [36]
Arduino DUE [5]	101,5 x 53,3	54 digitales y 12 analógicos	33,06 € [35]

Cuadro 9.1: Comparativa entre Arduinos

Como se puede observar en la tabla 9.1, tanto la placa Arduino MEGA como la DUE tienen pines de sobra pero son demasiado grandes y caras. Entre la UNO y la NANO se aprecia una diferencia de tamaño notable. Tanto el precio como los pines son similares y el número adecuado en ambos casos.

Para la conexión entre ambas controladoras se tomaron en cuenta dos alternativas:

- Conexión I2C
- Conexión SPI

Ambas alternativas son analizadas a continuación.

9.1.1. Conexión I2C (Inter-Integrated Circuit)

Esta conexión se realiza mediante el bus I2C, un estándar creado para facilitar el envío de información entre diferentes microcontroladores. Requiere solo dos cables, uno para el envío de la información (SDA) y otro para la señal de reloj (SCL). En este caso, al estar los módulos alimentados independientemente, es necesario un tercer cable para unir las tierras (GND) de ambas placas.

El sistema está basado en la arquitectura maestro-esclavo. Un dispositivo actúa como maestro, que inicia la conexión con el esclavo y envía o recibe los datos necesarios. En este caso, el maestro sería el módulo central y el esclavo el nuevo módulo externo.

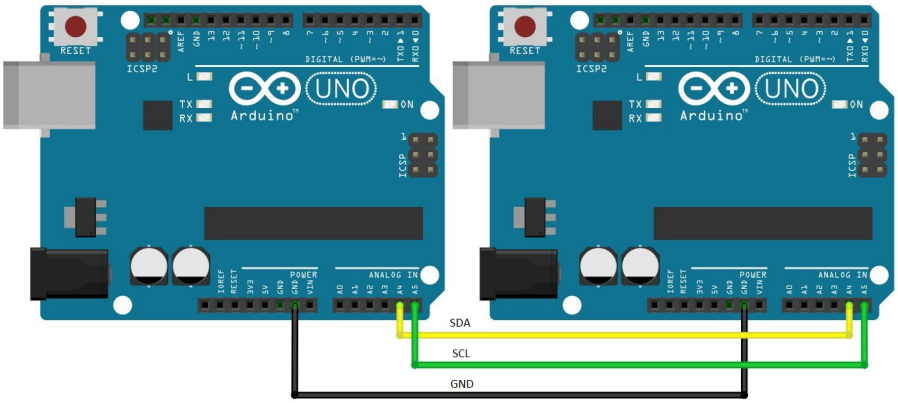


Figura 9.1: Diagrama de conexión I2C.

9.1.1.1. Ventajas

- Utiliza pocos cables.
- No importa la distancia que tenga que recorrer la información.
- Se puede verificar que la información llega correctamente.
- El dispositivo que recibe los datos no necesita conocer la longitud del mensaje.

9.1.1.2. Desventajas

- No permite enviar y recibir simultáneamente.
- No es la conexión más rápida.

9.1.2. Conexión SPI (Serial Peripheral Interface)

Es otro estándar creado también para facilitar el envío de información entre microcontroladores. Requiere tres cables para la conexión, uno para enviar la información (MOSI), otro para recibir la información (MISO) y el último para la señal de reloj (SCK). Además es necesario uno más por cada esclavo (SS) y en este caso también se debe unir ambas tierras (GND), por lo que serían necesarios cinco cables.

Este sistema también esta basado en la arquitectura maestro-esclavo. Al contrario que en el bus I2C, se usan dos líneas independientes para enviar y recibir información por lo que es posible realizar ambas acciones de forma simultánea.

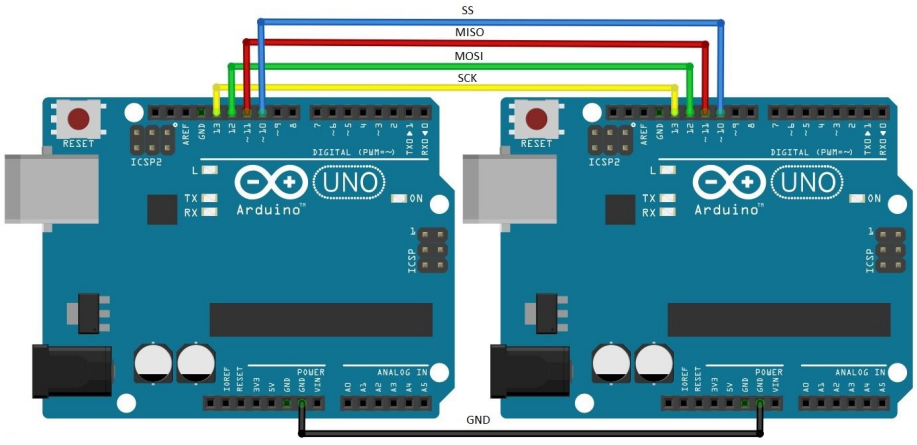


Figura 9.2: Diagrama de conexión SPI.

9.1.2.1. Ventajas

- Permite enviar y recibir información simultáneamente.
- Alta velocidad de transmisión.

9.1.2.2. Desventajas

- Requiere más cables, para este proyecto serían suficientes ya que se utilizarían cinco de los seis disponibles.
- Solo es adecuado para distancias muy cortas.
- No se puede verificar que la información llegue correctamente.
- Ambos dispositivos deben conocer la longitud de los mensajes enviados.

9.2. Elección de componentes

Este proyecto requiere el uso de varios componentes:

- Sensor de pH.
- Sensor de EC.
- Sensor de ORP.
- Sensor de Temperatura.
- Brújula magnética.
- GPS.
- Lector de tarjeta SD.

Para conectar los componentes a la placa Arduino se han estudiado dos posibilidades:

- Diseñar un circuito que permita la conexión de cada componente a Arduino.
- Utilizar componentes preparados para su funcionamiento en Arduino.

La opción más sencilla y segura es la segunda.

La brújula magnética, el GPS y el lector son facilmente de encontrar ya que son elementos muy usados en Arduino. Los otros sensores son más dificiles de encontrar ya que no son sensores de uso tan general.

9.3. Iluminación externa

Para el sistema de iluminación del módulo externo se decidió usar diferentes LEDs blancos de alta luminosidad conectados a él. Se han estudiado dos formas de conectarlos:

- Conexión directa al módulo.
- Conexión a la batería.

9.3.1. Conexión directa al módulo

En este tipo de conexión los LEDs se conectan directamente a los pines de Arduino. Su mayor ventaja es que al ser una conexión directa solo es necesario utilizar las resistencias necesarias para cada LED y los cables al Arduino.

Sin embargo, al estar alimentados por el propio Arduino consumen una gran cantidad de potencia. Esto plantea un problema, ya que Arduino no es capaz de proveer una gran cantidad de corriente y además, cada pin solo puede proporcionar un máximo de 20mAh. Por ejemplo, para 8 LEDs existirían varias opciones:

Conexión	Consumo	Pines	Luminosidad (aproximada)
Un LED	160 mA (8 x 20 mAh)	8	20 cd
Dos LEDs	80 mA (4 x 20 mAh)	4	10 cd
Cuatro LEDs	40 mA (2 x 20 mAh)	2	5 cd
Ocho LEDs	20 mA (1 x 20 mAh)	1	2,5cd

Cuadro 9.2: Comparativa de conexión

Como se observa en la tabla 9.2, a medida que se baja el consumo también baja la luminosidad de cada LED.

9.3.2. Conexión a la batería

Si se conectan directamente a la batería, se superan las limitaciones mencionadas anteriormente. En este caso se necesita usar un transistor que sirva como interruptor entre la batería y los LEDs ya que se quiere poder encender y apagar desde Arduino.

Dependiendo de la potencia de la batería elegida es posible conectar en serie y en paralelo los LEDs para un mayor ahorro de consumo sin perder luminosidad. Para el ejemplo anterior de 8 LEDs se puede realizar la siguiente conexión:

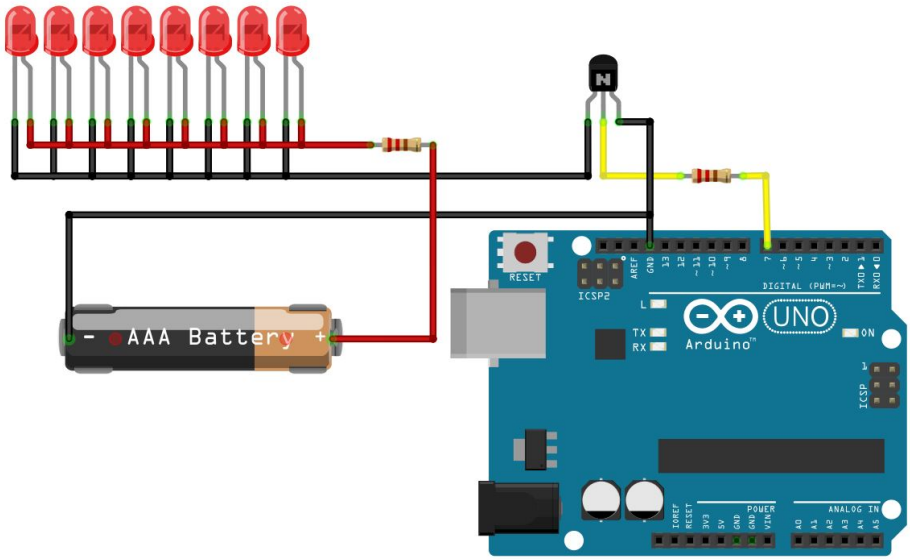


Figura 9.3: Diagrama de conexión de los LEDs a la batería.

10. Descripción de la Solución Propuesta

Una vez analizadas las diferentes alternativas para cada caso, es necesario elegir las mejores alternativas para la solución del proyecto.

10.1. Diseño de la arquitectura de la solución

En el diagrama de bloques de la figura 10.1 se observa como los sensores se conectan al módulo externo mediante un sistema que envía los valores de las variables actualizadas en cada momento. El módulo GPS y el giroscopio también se conectan al módulo externo enviando los valores necesarios para calcular la posición.

En el caso del módulo MicroSD y los LEDs es el módulo externo el que envía la información. En el primer caso para que guarde las variables en una tarjeta MicroSD y en el segundo para controlar el encendido y apagado de las luces.

El módulo externo se conecta al ROV, o módulo central, de forma bidireccional y envía la información que debe mostrarse en la interfaz y recibe órdenes, como encender y apagar las luces.

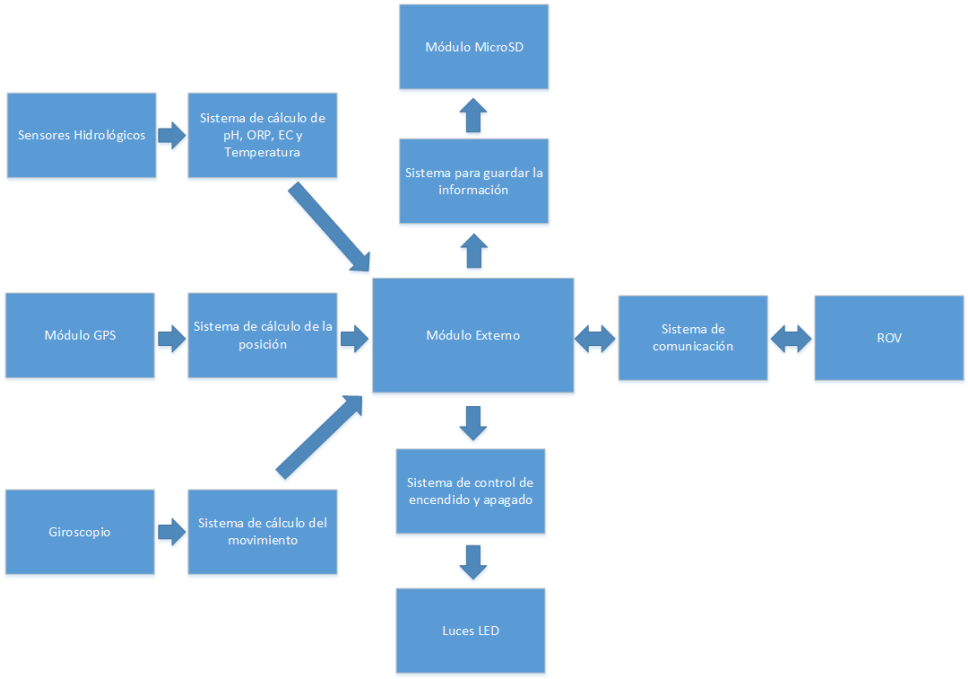


Figura 10.1: Diagrama de bloques del sistema.

En la figura 10.2 se puede apreciar el diseño final tras las consideraciones de los próximos apartados.

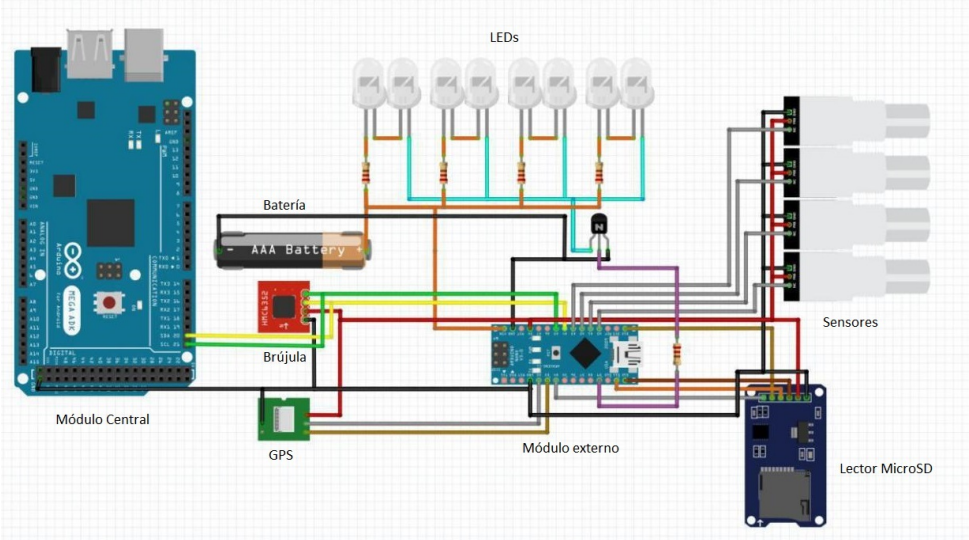


Figura 10.2: Diseño final de la arquitectura.

10.2. Elección de la plataforma del módulo externo

Como se observa en la tabla 9.1, Arduino NANO es el dispositivo más pequeño económico, además de tener pines suficientes para la conexión de todos los componentes necesarios.

Las placas MEGA y DUE son descartadas inmediatamente por su gran tamaño y la UNO porque, aunque sea pequeña, es considerablemente superior a la NANO.

Para la conexión entre ambos módulos se elige la conexión I2C. Su principal ventaja es que ocupa solo tres cables, dejando otros tres libres para ampliaciones futuras. Además permite verificar que la información llega correctamente entre ambos módulos.

La placa Arduino MEGA del módulo central actúa como maestro para así poder enviar órdenes al módulo externo cuando el usuario lo requiera, además podrá requerir las variables que quiera mostrar en la interfaz del ROV. La placa Arduino NANO realiza la función de esclavo y su labor es procesar toda la información que recibe de sus diversos componentes y tenerlos listos para enviar al maestro cuando se los requiera, además, cuando recibe la petición, podrá encender/apagar las luces y proceder al almacenamiento de los datos mostrados en la tarjeta MicroSD.

A continuación se muestra el código de comunicación de ambos:

10.2.1. Código del Maestro

El maestro del sistema es la placa Arduino MEGA del ROV, por lo que hay que modificar su código para implementar la comunicación I2C.

Al contrario que el comportamiento del software de una placa Arduino normal, este sistema de control no tiene un apartado de «setup» donde se puedan inicializar funciones de conexión como necesita el protocolo I2C. Para solucionar este problema, se crea una pequeña librería donde, a través de su constructor, es posible inicializar la conexión con el esclavo.

A continuación se muestran las funciones necesarias para la comunicación, el código completo de la librería puede consultarse en el Anexo 18.1.4.1.

```
#include "comi2c.h"
#include <Wire.h> //Libreria I2C.

//Constructor que recibe la ID del esclavo e inicia la comunicacion I2C.
comi2c::comi2c(byte id){
  addr=id;  //Se asigna el identificador del esclavo para su posterior uso.
}

//Funcion que desempaqueta la respuesta para transformarla en un entero.
int comi2c::receiveResponse(){
  Wire.beginTransmission(addr); //Se inicia la transmision con el esclavo.
  int receivedValue = Wire.read() << 8 | Wire.read();   //Desempaqueta la
    respuesta.
  Wire.endTransmission(true);   //Finaliza la transmision.
  return receivedValue;
}

//Funcion que envia la orden al esclavo. En este ejemplo solo tiene un caso con
    0 (ph).
float comi2c::request(String op){
  int type=0;   //Existen dos tipos dependiendo del tipo de dato. 1 = int, 0 =
    float.
  byte DataPacket[1];
  if(op=="ph"){
    DataPacket[0] = 0;
    type=1;
  }
  Wire.beginTransmission(addr); //Inicia la transmision para enviar el comando.
  Wire.write(DataPacket, 1);    //Envia el comando.
  Wire.endTransmission(true);   //Finaliza la transmision.
  delay(10);
  float response = receiveResponse();   //Cuando la respuesta es desempaquetada
     se recibe.

  //Dependiendo del tipo de dato se realiza la siguiente operacion.
  if(type==0){
    return response;
  }else return response/100;
}
```

Para iniciar la comunicación desde el software de control del ROV es necesario crear el constructor y llamar a la función «request». En el anexo 18.2.1 puede consultarse el código del maestro.

10.2.2. Código del Esclavo

En este código se implementa el esquema del proceso de comunicación de la placa Arduino NANO con la placa maestra. El código completo, que será descrito en el apartado 10.5.2, puede consultarse en el Anexo 18.2.2.

```
1
2 #include <Wire.h> //Libreria I2C.
3
4 const byte SlaveDeviceId = 8; //ID del esclavo para la conexion I2C.
5 byte LastMasterCommand = 0; //Variable donde se guarda el comando enviado por
      el maestro.
6
7 void setup(){
8     Wire.begin(SlaveDeviceId);  //Se inicia la conexion I2C con su ID.
9     Wire.onReceive(receiveDataPacket); //Funcion para recibir peticiones I2C.
10    Wire.onRequest(slavesRespond);  //Funcion para responder por I2C.
11 }
12
13 //Funcion que recibe el comando de la accion a ejecutar del maestro.
14 void receiveDataPacket(int howMany){
15   LastMasterCommand = Wire.read();
16 }
17
18 //Funcion para responder al maestro con el dato solicitado.
19 void slavesRespond(){
20   int returnValue = 0; //Variable que se va a enviar al maestro como respuesta.
21
22   //Switch encargado de guardar en la variable el valor deseado. En este
       ejemplo solo tiene un caso con 0.
23   switch(LastMasterCommand){
24     case 0:
25       returnValue = 0;
26       break;
27   }
28
29   //Se divide el dato en un buffer de dos bytes para poder enviar valores
       negativos y mayores a 255 a traves de I2C.
30   byte buffer[2];
31   buffer[0] = returnValue >> 8;
32   buffer[1] = returnValue & 255;
33   Wire.write(buffer, 2); //Se envia el buffer al maestro.
34 }
```

10.3. Elección de los componentes

La mejor opción es la compra de los componentes diseñados para interactuar con Arduino. Estos deben ser adquiridos en empresas especializadas.

A continuación se detalla cada uno de los componentes y su forma de conexión al módulo externo:

10.3.1. Sensores

Los sensores seleccionados han sido fabricados por la compañía DFRobot [13] [14] [15]. Estos sensores vienen acompañados de pequeña placa controladora, que dispone de un conector para el sensor y tres pines compatibles con Arduino.

En la figura 10.3 se puede ver un ejemplo de conexión:

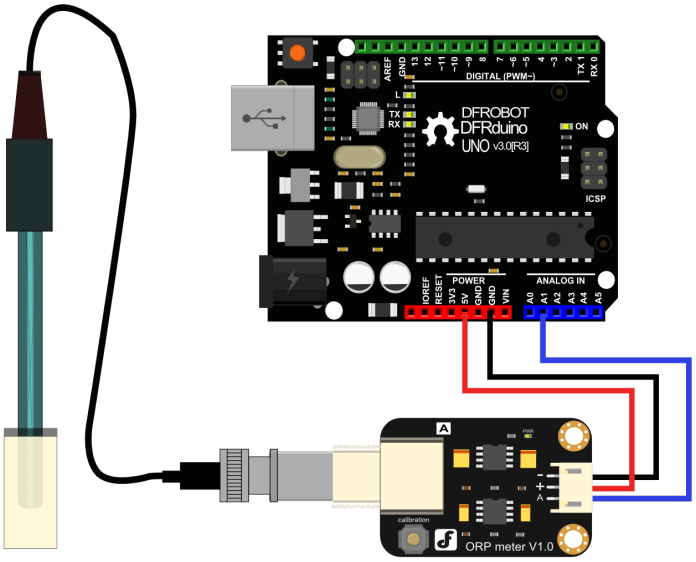


Figura 10.3: Conexión de los sensores.

Uno de los pines se debe conectar al pin de alimentación de Arduino (5V), otro a tierra (GND) y el tercero a un pin analógico para la señal.

Los cuatro sensores funcionan exactamente igual, por lo que se pueden unir las señales 5V y GND en una, suponiendo un ahorro de cableado. De esta forma quedan cuatro cables de señal analógica, GND y 5V. Se liberan la mitad de cables (de doce a seis).

Cada uno de los sensores tiene un manual para calibración y una documentación básica relativa a su funcionamiento con Arduino. Con esta documentación se crea una librería por cada sensor que ser consultada en el anexo 18.1.

10.3.2. Módulo GPS

Se utiliza un módulo GPS GY-NEO6MV2 [16], que es ampliamente utilizado en drones aéreos y el preferido por los usuarios de Arduino. Su elección se basa en la completa documentación que existe de este módulo y la gran comunidad en foros y páginas de internet que lo respaldan.

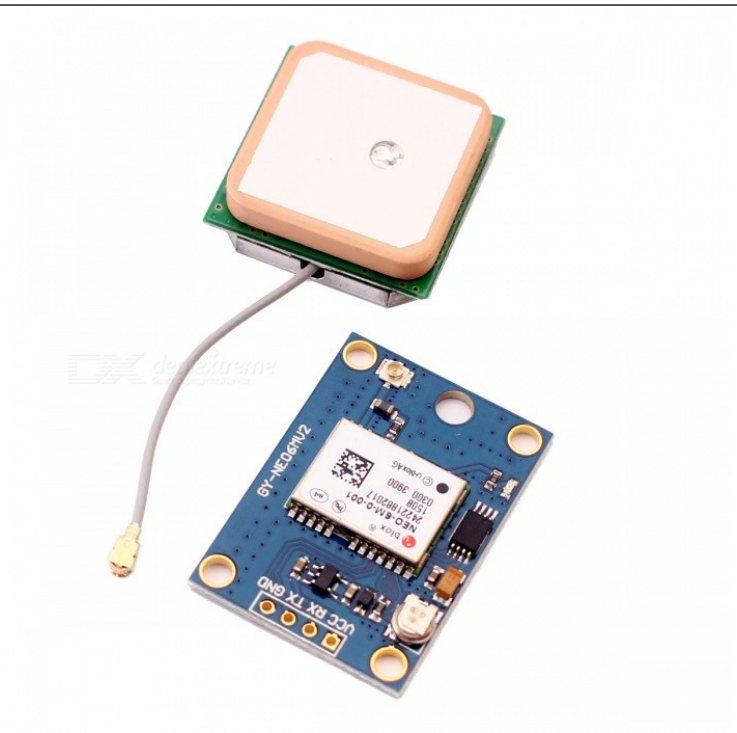


Figura 10.4: Módulo GPS GY-NEO6MV2.

Este módulo GPS trae incorporada una antena cerámica para mejorar la señal y cuatro pines de conexión, GND, 5V y 2 pines de conexión serial (TX y RX). Su conexión a Arduino se realiza del siguiente modo:

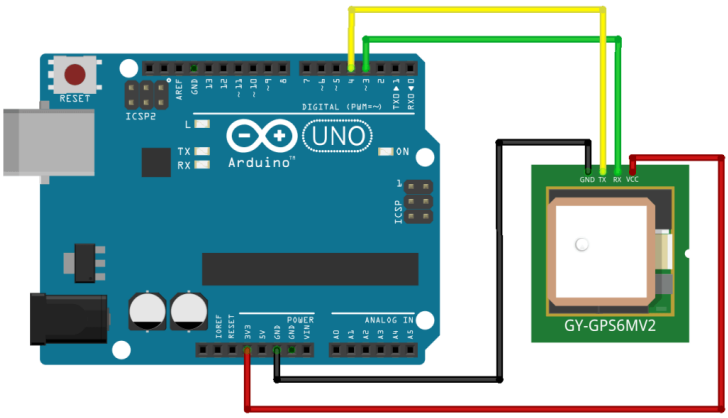


Figura 10.5: Conexión del módulo GPS GY-NEO6MV2.

Siendo los pines RX y TX totalmente configurables.

10.3.3. Sensor magnético

En este caso se ha optado por el sensor GY-273 con chip HMC5883L [19] teniendo en consideración los mismos aspectos que en el módulo GPS.

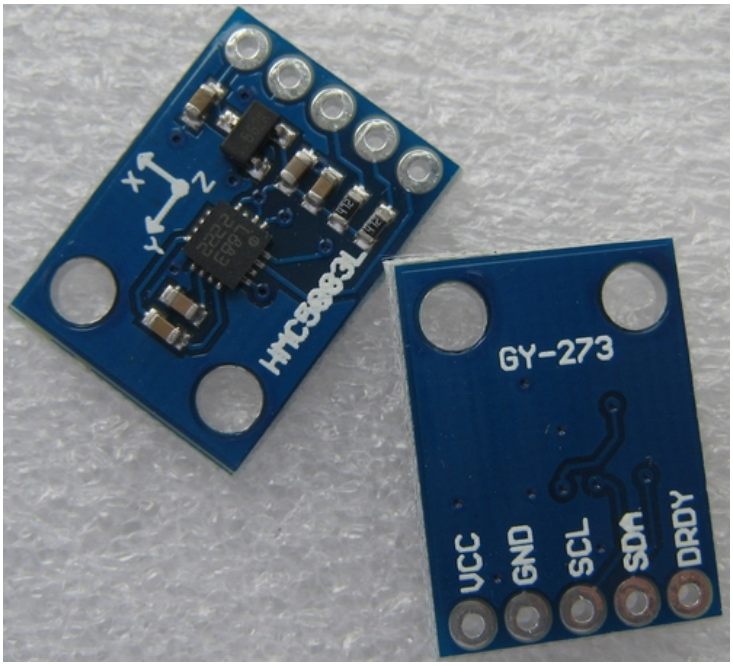


Figura 10.6: Brújula GY-273.

Este componente trae incorporado el chip HMC5883L, un giroscopio de tres ejes y cinco pines de los que solo necesitamos cuatro para su conexión a Arduino. La conexión se realiza mediante el protocolo I2C, por lo que son necesarios 5V, GND, SDA y SCL.

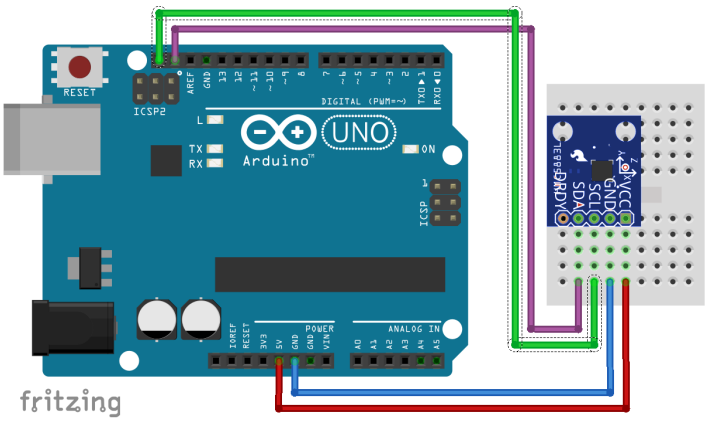


Figura 10.7: Conexión brújula GY-273.

10.3.4. Lector MicroSD

Para este componente se elige un componente genérico [18], ya que se trata del componente más básico y simple. El lector trae un conector para microSD y seis pines (5V, GND, MISO, MOSI, SCK y CS) necesarios para la conexión mediante el protocolo ISP a Arduino.

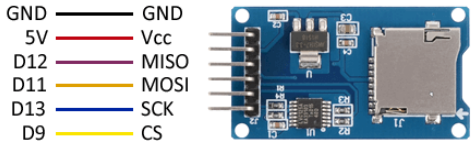


Figura 10.8: Conexión lector microSD.

10.4. Elección de la iluminación y alimentación

Tras el estudio de las diferentes propuestas se decide usar iluminación alimentada por la batería ya que el número de componentes conectados a la placa Arduino es elevado. Así se evita sobrecargar la placa y podemos disponer de la máxima luminosidad. Se opta por usar ocho LEDs (cuatro en cada lado del ROV) y crear una pequeña placa PCB donde se puedan soldar las resistencias necesarias y el transistor, así como los pines que saldrán hacia la batería y la placa Arduino. Los LEDs se distribuyen cuatro a cada lado de la carcasa.

Antes de decidir el esquema final de los LEDs y sus resistencias es necesario elegir la batería con la que se va a alimentar el módulo externo. Se selecciona una batería de pequeño tamaño para poder integrarla en la estructura y que su voltaje se encuentre entre 7V y 9V, que es el adecuado para alimentar una placa Arduino por su pin VIN. La batería elegida es una batería de tipo LIPO de 7,4V y 2800mAh [3].

Figura 10.9: Batería LIPO 7,4V y 2800mAh.

Llegados a este punto, se deben realizar unos cálculos para conseguir que los ocho LEDs se distribuyan de la manera más eficiente consumiendo lo mínimo posible para dar el máximo de luminosidad.

Para este diseño se va a emplear un circuito mixto, mezclando serie y paralelo. En primer lugar, se busca colocar los LEDs en serie. Un circuito en serie comparte la misma intensidad (I) y se va sumando el voltaje necesario para componente (V), por lo que mientras más LEDs se puedan colocar en serie mayor será el ahorro de intensidad.

$$I = I^1 = I^2 = ... = I^n$$
$$V = V^1 + V^2 + ... + V^n$$

Los LEDs usados son de alta luminosidad con una intensidad de 20mAh y una corriente de 3,4V. Al usar una batería de 7,4V se observa que se puede usar un circuito en serie dos a dos con una intensidad de 20mAh y un voltaje de 6,8V.

$$I = 20mAh$$
$$V = 3,4 + 3,4 = 6,8V$$

El segundo paso es colocar las parejas de LEDs en paralelo, en un circuito en paralelo se comparte el mismo voltaje y se va sumando la intensidad.

$$I = I^1 + I^2 + ... + I^n$$
$$V = V^1 = V^2 = ... = V^n$$

Se colocan las cuatro parejas de LEDs en paralelo obteniendo una intensidad total de 80mAh y un voltaje de 6,8V.

$$I = 20 + 20 + 20 + 20 = 80mAh$$
$$V = 6,8V$$

Una vez el diseño del circuito está completo hay que calcular el valor de las resistencias (R) necesarias para cada par de LEDs, para ello se utiliza la ley de Ohm en el circuito en serie.

$$V = R * I$$

De la que se obtiene:

$$R = \frac{V}{I}$$

Donde V es la diferencia de voltaje entre el voltaje de la batería y el requerido por los LEDs e I la intensidad de los LEDs.

$$R = \frac{7,4 - 6,8}{0,02} = \frac{0,6}{0,02} = 30\Omega$$

Con este resultado se debe buscar la resistencia comercial más próxima, en este caso 33Ω.

Ahora se deben calcular la potencia de la resistencia mediante la siguiente fórmula:

$$P = V * I$$

En este caso, al usar una resistencia un poco superior a la ideal, se debe volver a calcular la intensidad que pasa por el circuito.

$$I = \frac{V}{R} = \frac{0,6}{33} = 0,18 = 18mAh$$

Se calcula ahora la potencia de la resistencia con la fórmula anterior.

$$P = V * I = 0,6 * 0,18 = 0,108W$$

El valor es inferior a 0,125W por lo que se pueden usar las resistencias de 1/8W. El circuito final queda de la siguiente manera:

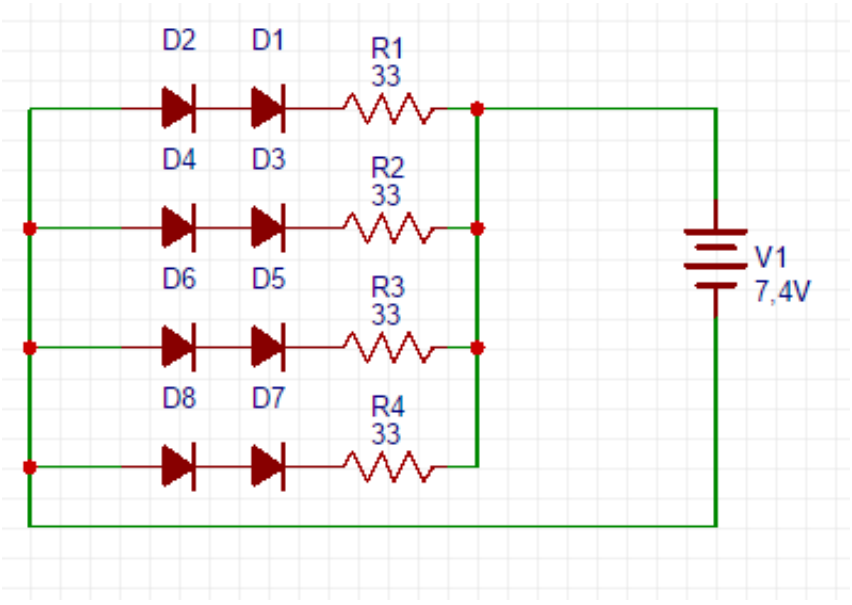


Figura 10.10: Circuito de LEDs.

Con el circuito de los LEDs completo se debe buscar la manera de conectarlos a Arduino para que actúe de interruptor y permitir al usuario encender y apagar las luces. Para esto, se usa un transistor NPN en modo interruptor.

Un transistor NPN tiene tres patillas, colector (c), emisor (e) y base (b). El transistor se coloca en mitad del circuito, la patilla del colector recibiendo la señal de la batería y la patilla del emisor hacia el resto del circuito.

La base se conecta a la placa Arduino mediante una pequeña resistencia de seguridad por si enviase una señal demasiado alta.

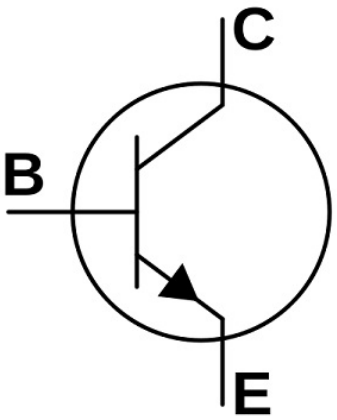


Figura 10.11: Patillas del transistor NPN.

El circuito permanece abierto hasta que recibe una señal del Arduino por la base, en ese momento se cierra el circuito y las luces pueden encenderse.

Para este proyecto se va a usar el transistor 2N2222, transistor ampliamente utilizado para proyectos Arduino que garantiza 500mAh y 50V, más que suficiente.

Una vez todos los componentes elegidos y cálculos realizados, se diseña una pequeña PCB con el fin de ubicar el transistor, la resistencia de la base y los pines en un mismo componente.

10.4.1. Diseño PCB para los LEDs

La PCB es de pequeño tamaño para poderla colocar en los tubos de la carcasa, en ella se colocan nueve pines, el transistor y una pequeña resistencia de 100Ω.

Los nueve pines se dividen de la siguiente manera:

- Cuatro pines para los LEDs. Dos por cada lado de LEDS, izquierdo y derecho.
- Tres pines para conectar la placa Arduino, VIN, GND y un pin digital para el control de los LEDs.
- Dos pines para conectar la batería, GND y 5V.

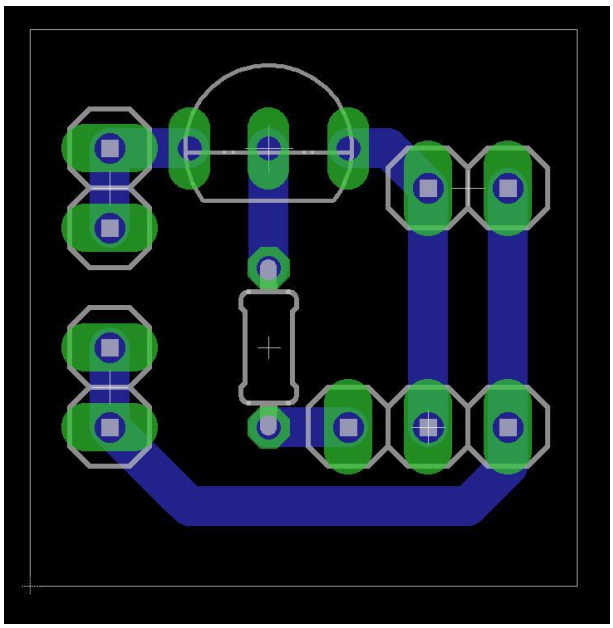


Figura 10.12: Diseño PCB para la conexión.

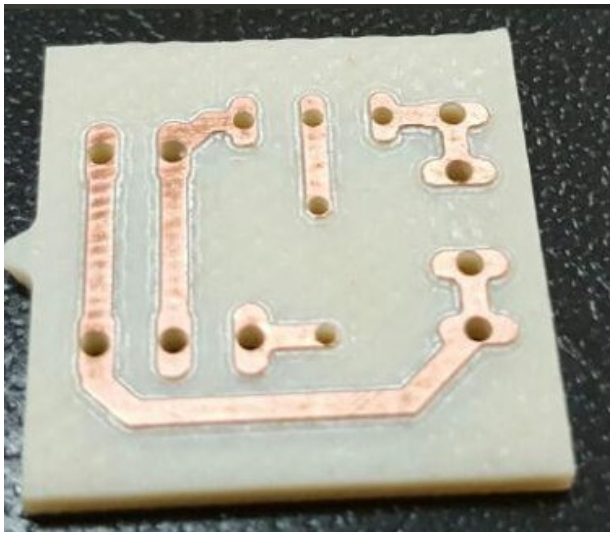


Figura 10.13: PCB para la conexión.

10.5. Diseño del módulo externo

Una vez estudiadas y seleccionadas todas las alternativas se diseña el módulo externo, su sistema electrónico y su software. Consta de los siguientes elementos:

- Arduino NANO.
- Sensor de pH.
- Sensor de ORP.
- Sensor de EC.
- Sensor de temperatura.
- Sensor magnético.
- GPS.
- Lector MicroSD.

Para la conexión de estos elementos se diseña una placa PCB de pequeño tamaño. Debe permitir la conexión de los diferentes componentes y el ROV.

Para la conexión al ROV se emplea el protocolo I2C y su arquitectura maestro-esclavo. Para ello se debe desarrollar el software necesario que permita al módulo externo recopilar toda la información necesaria, almacenarla físicamente y enviarla al ROV cuando lo requiera.

10.5.1. Diseño PCB

El diseño final se puede ver en la figura 10.2, en él se aprecia la gran cantidad de conexiones que se deben realizar por lo que el disponer de una PCB donde conectar directamente los cables de cada componente supone una gran ventaja.

Para el diseño se deben tener en cuenta los requisitos por parte de cada componente, ya que algunos hacen uso de pines específicos, como los usados por el protocolo ISP y es necesario reservarlos. Otros como los usados por el protocolo I2C deben ser reservados pero pueden ser usados para conectar hasta 128 dispositivos.

Para esto, se deben estudiar primero los pines de Arduino NANO:

- Pines digitales 0-13.
  - Pin 0: RX.
  - Pin 1: TX.
  - Pines 2 y 3: Interrupciones.
  - Pines 3, 5, 6, 9, 10, 11: PWM.
  - Pin 10: SS.
  - Pin 11: MOSI.
  - Pin 12: MISO.
  - Pin 13: SCK.
- Pines analógicos A0-A5.
  - Pin A4: SDA.
  - Pin A5: SCL.

Una vez reservados los pines necesarios para algunos componentes, se empieza a realizar el diseño teniendo en consideración los pines finales que usará cada componente:

- Arduino MEGA (conexión entre módulo central y módulo externo):
  - GND: GND.
  - SDA: A4.
  - SCL: A5.
- Sensor pH:
  - GND: GND.
  - 5V: 5V.
  - Señal analógica: A0.
- Sensor ORP:
  - GND: GND.
  - 5V: 5V.
  - Señal analógica: A1.
- Sensor EC:
  - GND: GND.
  - 5V: 5V.
  - Señal analógica: A2.

- Sensor de temperatura:
  - GND: GND.
  - 5V: 5V.
  - Señal analógica: A3.
- Sensor magnético:
  - GND: GND.
  - 5V: 5V.
  - SDA: A4.
  - SCL: A5.
- Módulo GPS:
  - GND: GND.
  - 5V: 5V.
  - RX: 3.
  - TX: 2.
- Lector microSD
  - GND: GND.
  - 5V: 5V.
  - MISO: 12.
  - MOSI: 11.
  - SCK: 13.
  - CS: 4.
- PCB LEDs:
  - GND: GND.
  - Alimentación: VIN.
  - Señal digital: 9.

El diseño de la PCB final se puede observar en la figura 10.14.

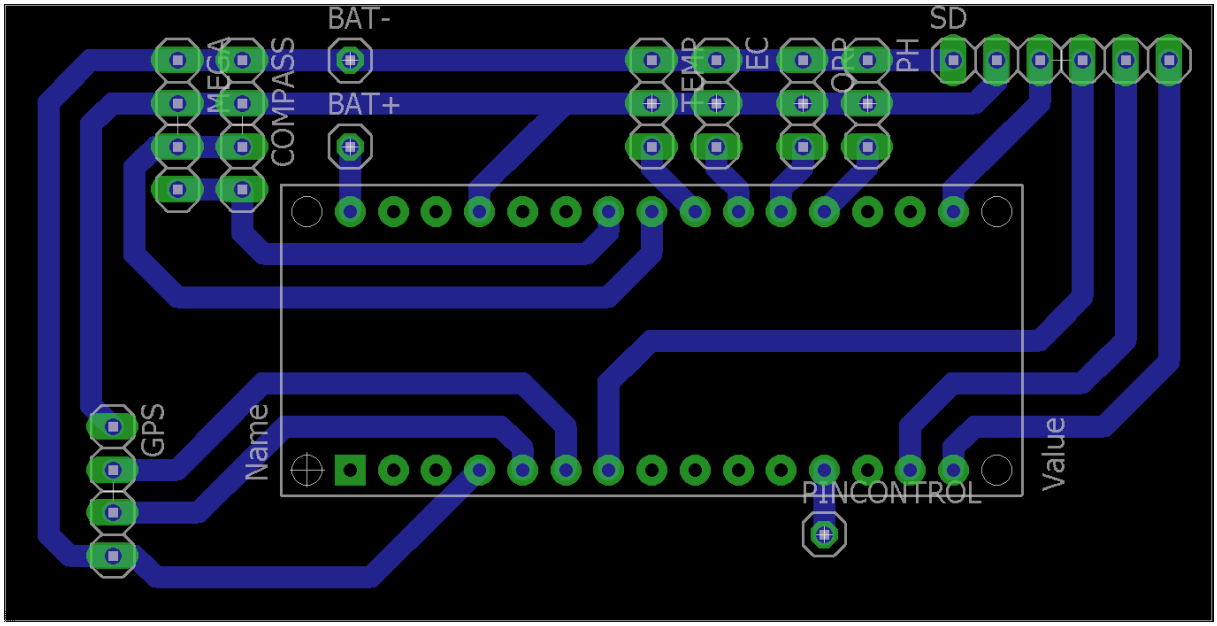


Figura 10.14: Diseño PCB.

10.5.2. Diseño del software del módulo externo

El software del módulo externo se divide en tres fases bien diferenciadas, descritas a continuación:

10.5.2.1. Recopilación de datos

Esta primera fase es la encargada de recopilar continuamente datos de los sensores. Su objetivo es que las variables estén siempre actualizadas con su último valor muestreado para enviarlos cuando el maestro los requiera.

Para los sensores hidrológicos se emplean las librerías descritas en el Anexo 18.1 y para el sensor magnético la librería «HMC5883LSimple» [23], creada por la comunidad de Arduino y la librería I2C (wire) [12].

```
#include <sensorORP.h>
#include <sensorpH.h>
#include <sensorEC.h>
#include <HMC5883L_Simple.h>
#include <Wire.h>

int valorORP, valorEC, valorCompass;   //Variables para los sensores.
float valorpH, valorTemp;

```

```
10 //Constructores de los objetos sensores pasandole los pines al que estan
       conectados.
11 sensorORP sensor_orp(A2);
12 sensorEC sensor_ec(A1);
13 sensorpH sensor_ph(A0);
14 HMC5883L_Simple Compass;
15
16 void setup(){
17     //Inicializacion de los sensores.
18     sensor_orp.setup();
19     sensor_ec.setup();
20     sensor_ph.setup();
21     //Opciones para calibrar el compass y muestre el verdadero norte.
22     Compass.SetDeclination(23, 35, 'E');
23     Compass.SetSamplingMode(COMPASS_SINGLE);
24     Compass.SetScale(COMPASS_SCALE_130);
25     Compass.SetOrientation(COMPASS_HORIZONTAL_X_NORTH);
26
27 }
28
29 void loop(){
30   delay(100); //Delay de seguridad.
31
32   //Funciones de las librerias de sensores para obtener los valores y tenerlos
       listos en caso de que el maestro los pida.
33   valorORP = sensor_orp.getORP();
34   valorpH = sensor_ph.getpH();
35   sensor_ec.getVal();
36   valorEC = sensor_ec.getEC();
37   valorTemp = sensor_ec.getTemp();
38   valorCompass=Compass.GetHeadingDegrees();
39 }
```

10.5.2.2. Almacenamiento de datos

A medida que el bucle de recopilación se repite es necesario almacenar las variables en un fichero a modo de historial para su estudio.

Se emplea la librería «SD» [10] que incorpora Arduino para el manejo de ficheros en tarjetas extraíbles y la librería «SoftwareSerial» [11] para la lectura de datos, necesaria para el GPS. Además, se emplea una librería creada por la comunidad de Arduino «TinyGPS++» [25] para recibir información de los satélites.

Se emplea una variable de control que cada 30 segundos almacena los datos en una tarjeta microSD. Para esto, se utiliza la función «millis()» [4] de Arduino que devuelve el tiempo que lleva encendida la placa.

Los datos que se van a almacenar son todos los recopilados por los sensores y la posición del GPS.

```
1 #include <SD.h> //Libreria para manejar la SD.
2 //Librerias para el GPS.
3 #include <TinyGPS++.h>
4 #include <SoftwareSerial.h>
5
```

```
6  const byte pinSD = 4;  //Variable con el pin al que se conecta la SD.
7  //Contador utilizado para el guardado en la SD.
8  long intervalo = 3000;
9  long tiempo = 0;
10 long tiempoAnterior = 0;;
11
12 //Constructores para el GPS.
13 TinyGPSPlus gps;
14 SoftwareSerial ss(3, 2);
15
16 float valorLongitud, valorLatitud; //Variables para el GPS.
17 File myFile;  //Variable para el archivo de la SD.
18
19 void setup{
20     ss.begin(4800);      //Inicio del GPS.
21     SD.begin(pinSD);     //Inicio de la SD.
22
23 }
24
25 void loop{
26     tiempo = millis();
27
28     if(tiempo - tiempoAnterior > intervalo) {
29         tiempoAnterior = tiempo;
30         //Si la comunicaciĂłn con el GPS esta disponible se reciben los datos y
       se guardan.
31         if (ss.available() > 0){
32             valorLongitud = gps.location.lng();
33             valorLatitud = gps.location.lat();
34         }
35
36         //Se abre el archivo y se guarda la informacion.
37         myFile = SD.open("info.txt", FILE_WRITE);
38         if (myFile) {
39             myFile.print("PosiciĂłn: ");
40             myFile.print(valorLongitud);
41             myFile.print(", ");
42             myFile.print(valorLatitud);
43             myFile.print(" - pH: ");
44             myFile.print(valorpH);
45             myFile.print(" - ORP: ");
46             myFile.print(valorORP);
47             myFile.print("mV - EC: ");
48             myFile.print(valorEC);
49             myFile.print("ms - Temperatura: ");
50             myFile.print(valorTemp);
51             myFile.println(" ÂşC");
52         }
53         myFile.close();
54     }
55 }
```

10.5.2.3. Comunicación

La comunicación se encarga de escuchar continuamente al maestro con el objetivo de enviar los datos requeridos, encender y apagar el sistema de iluminación.

Se emplea el código descrito en el apartado 10.2.2 modificando la función «switch» con todos los casos que pueden ocurrir.

```
1
2   //Switch encargado de guardar en la variable el valor deseado para enviarla
      posteriormente.
3   //0 = pH
4   //1 = ORP
5   //2 = EC
6   //3 = Temperatura
7   //4 = Brujula
8   //Y de realizar las siguientes funciones.
9   //10 = Encender las luces
10  //11 = Apagar las luces
11  switch(LastMasterCommand){
12    case 0:
13      returnValue = valorpH*100;
14      break;
15    case 1:
16      returnValue = valorORP;
17      break;
18    case 2:
19      returnValue = valorEC*100;
20      break;
21    case 3:
22      returnValue = valorTemp*100;
23      break;
24    case 4:
25      returnValue = valorCompass;
26      break;
27    case 10:
28      digitalWrite(9, HIGH);
29      break;
30    case 11:
31      digitalWrite(9, LOW);
32      break;
33  }
```

10.6. Modificación de la plataforma OpenROV

Para la integración del sistema se deben modificar dos aspectos de la plataforma OpenROV. Esto se puede realizar gracias a que la plataforma es software y hardware libre. La documentación se encuentra disponible en su GitHub oficial [24].

En primer lugar, se debe modificar la interfaz de usuario original del ROV, que es muy caótica, pequeña y con una gran cantidad de datos que no son de interés para el usuario en este proyecto.

Se creará un nuevo plugin que pueda ser activado y desactivado. Y, en segundo lugar, se deben modificar la funcionalidad de forma que permita añadir nuevas funciones y opciones de manera sencilla, ya que, por defecto, solo admite las funciones que trae el software original.

Todos los cambios de realizan mediante el sistema de plugins, sistema que utiliza el software de OpenROV y permite activar y desactivar las funcionalidades desde la propia interfaz. Los plugins deben estar realizados en JavaScript. Se debe seguir la documentación oficial [33] para su buen funcionamiento y compatibilidad.

Para añadir un nuevo plugin al ROV se puede entrar a través de Cloud9, un entorno de desarrollo que incorpora OpenROV pero además de ser un sistema muy lento puede provocar errores en ocasiones. La segunda opción es entrar a través del protocolo FTP (File Transfer Protocol).

Este protocolo permite el intercambio de archivos basándose en la arquitectura cliente-servidor. Se emplea el programa «WinSCP» y a través de la dirección IP del ROV y el puerto 22 (puerto por defecto FTP) se accede al directorio de ficheros del servidor.

Una vez dentro del directorio se debe acceder a la dirección «/opt/openrov/cockpit/src/plugins» y crear una carpeta nueva con los archivos del nuevo plugin. Una vez hecho, se debe reiniciar el ROV y en el menú «Settings» de la interfaz aparecerá el nuevo plugin, permitiendo su activación y desactivación.

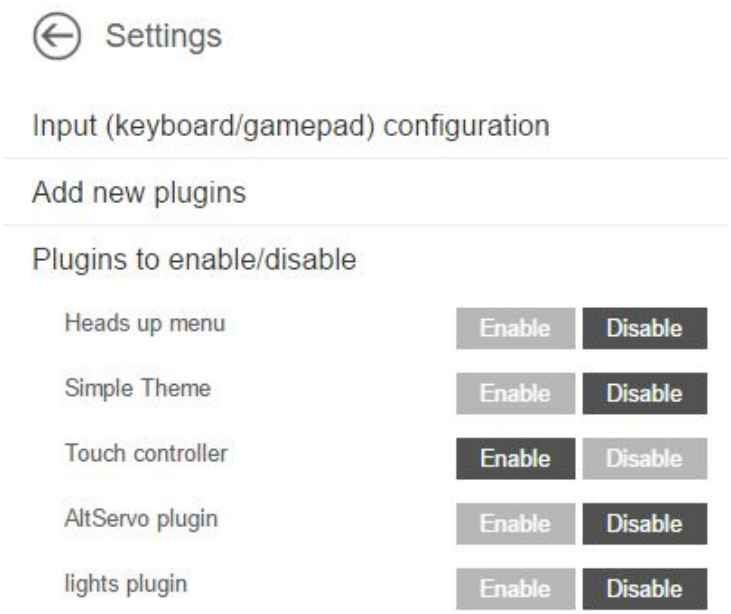


Figura 10.15: Menú de plugins.

10.6.1. Modificación de la interfaz

El principal inconveniente que tiene la interfaz por defecto de la plataforma es su gran cantidad de información distribuida por toda la pantalla, provocando que la zona donde aparece la imagen de la cámara sea relativamente pequeña.

Figura 10.16: Interfaz predeterminada.

Para suplir este problema, un usuario de la comunidad de OpenROV diseñó un plugin llamado SimpleTheme [22] que se encarga de quitar absolutamente toda la información de pantalla y ampliar la imagen de la cámara.

Al igual que el resto de la plataforma, este plugin es de código abierto por lo que se puede modificar su código CSS de acuerdo a las preferencias de cada usuario.

En este caso, se han superpuesto en el borde de la pantalla las opciones necesarias para el funcionamiento del sistema encima de la imagen procedente de la cámara, intentando que ocupe elespacio mínimo posible. El archivo CSS y el resto de los códigos del plugin están disponible en la página de GitHub del proyecto (apartado 5.4.1) y en el CD adjunto.

Además, se ha creado una pequeña tabla en el lado derecho con la información procedente e los sensores. Esta tabla solo es visible cuando se desliza el puntero del ratón por la derecha por lo que el resto del tiempo permanece oculta ayudando a una mejor imagen.

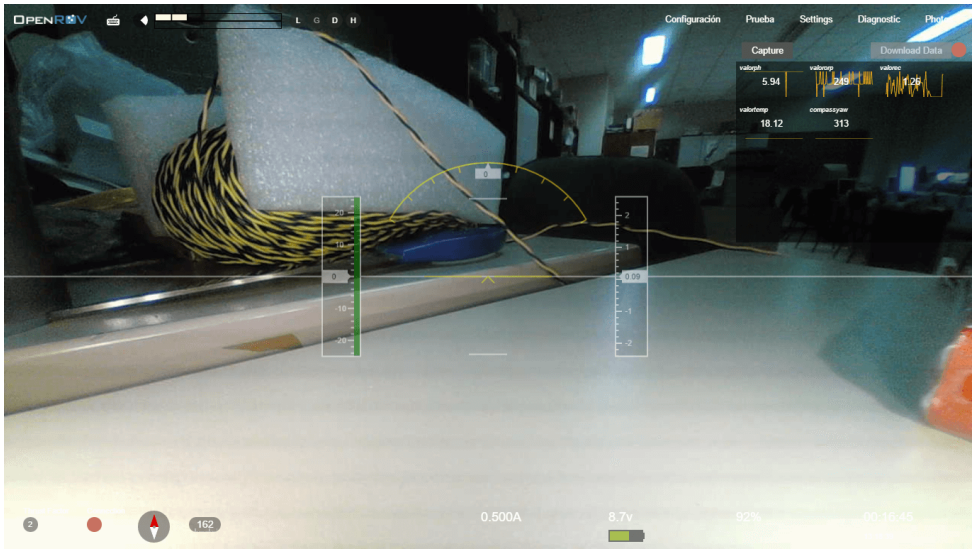


Figura 10.17: Interfaz nueva.

Se ha creado un nuevo menú en la parte superior de la interfaz con el nombre de «autogiro». Dicho menú despliega una pequeña rueda con diferentes grados y una casilla en blanco donde se puede escribir cualquier grado entre 1 y 360 grados. Este menú permite al ROV mantenerse siempre en una orientación dada.

Figura 10.18: Menú autogiro.

10.6.2. Modificación de la funcionalidad

La plataforma funciona mediante Node.js [32], que es un intérprete JavaScript incluido la tarjeta BeagleBone. El código principal del sistema está continuamente escuchando posibles eventos mediante sockets [44], cuando recibe un nuevo evento lo ejecuta de forma asíncrona mientras sigue escuchando y ejecutando otros.

Los eventos llegan a través de los diferentes plugins instalados y activados en el sistema. Dependiendo del tipo de evento lo ejecuta la propia BeagleBone o se envía a la Arduino MEGA para modificar el comportamiento de algún componente.

Para crear una nueva funcionalidad se debe crear un nuevo evento a escuchar, para esto se modifica el código principal del sistema, ubicado en la ruta «opt/openrov/cockpit/src/cockpit.js».

El nuevo evento escucha por un evento llamado «cmd_update» que recibe dos variables.

- cmd: Nombre del comando.
- var: Variable de valor numérico en caso de ser necesaria.

```
socket.on('cmd_update', function (data) {
    var cmd = data.cmd;
    var val = data.val;
    controller.send(cmd+'('+val+')');
  });
```

El plugin nuevo que va a enviar los eventos se crea siguiendo las normas de OpenROV [24] para preservar la compatibilidad con el resto del sistema.

Las dos funciones nuevas creadas se encargan de enviar al módulo externo la orden de encender o apagar las luces y de girar el ROV a unos grados determinados por el usuario.

```
  prueba.prototype.ledsaux = function ledsaux(valor) {
    this.cockpit.socket.emit('cmd_update', {cmd:"ledaux",val:valor});
  };
  prueba.prototype.autogiro = function autogiro() {
    this.cockpit.socket.emit('cmd_update', {cmd:"holdHeading_toggle",val:
    ruedaGrados});
  };
```

Estas funciones son llamadas desde el emisor de eventos. Para ello, se modifica y se crean tres nuevas llamadas.

```
{
  name: "prueba.leds",
  description: "On/Off LEDs",
  defaults: { keyboard: '9' },
  down: function () {
    if (ledaux==0){
```

```
7       rov.ledsaux(0);
8       ledaux=1;
9       }
10    else{
11      rov.ledsaux(1);
12      ledaux=0;
13    }
14    }
15  },
16  {
17    name: "prueba.grados",
18    description: "Cambia el valor de los grados",
19    defaults: { keyboard: '6' },
20    down: function () {
21      rov.autogiro();
22    }
23  },
24  {
25    name: "prueba.auto",
26    description: "Recta automĂĄtica",
27    defaults: { keyboard: '7' },
28    down: function () {
29      if (automotor==0){
30        rov.cockpit.emit('rovpilot.setThrottle', 1);
31        automotor=1;
32      }else{
33        rov.cockpit.emit('rovpilot.setThrottle', 0);
34        automotor=0;
35      }
36    }
37  },
```

La primera de ellas, «prueba.leds» se encarga de modificar el comportamiento del sistema de iluminación externo. Cuando se pulsa el número 9 del teclado se activan o desactivan, mandando el evento a las funciones previas.

La llamada «prueba.grados» modifica el valor de los grados en el que el ROV debe mantenerse y envía el evento a su función correspondiente «rov.autogiro()».

La última llamada utiliza un evento ya existente en el propio sistema utilizado para la activación de motores. Cuando se pulsa el número 7 del teclado el ROV activa el modo automático y sigue en línea recta hasta que vuelva a ser pulsado. Este modo es interesante cuando se combina al giro automático de grados ya que permite que el ROV siga un rumbo fijo aunque sea movido por oleadas o elementos externos, automáticamente va a volver a posicionarse.

Una vez el plugin está listo, se debe crear una nueva funcionalidad en el código Arduino. Para ello debe crearse una nueva librería en la ruta «opt/openrov/arduino/OpenROV» siguiendo de nuevo las normas establecidas en su documentación oficial.

Este código se va a encargar de comunicarse con el módulo externo (tomando el papel del maestro en la arquitectura maestro-esclavo) y modificar el valor de la brújula por defecto usando el proporcionado por la nueva.

El cambio de la brújula se debe a que la que viene suministrada por OpenROV genera un pequeño error que se va incrementando a lo largo del tiempo. Este error consiste en un incremento progresivo de los grados cuando el ROV se encuentra en estado de reposo, por lo que al cabo de 10 minutos sus grados pueden haber cambiado completamente (donde antes estaba el grado 0, ahora puede estar el grado 30). Otro error que tiene la brújula es mostrar datos completamente diferentes aleatoriamente, volviendo de nuevo al valor real.

A través de la librería de comunicación que se puede consultar en el Anexo 18.1.4.1 se realiza la llamada a su función «request» con los valores que necesitamos para la interfaz (pH, ORP, EC y brújula) y se envía para su impresión en la interfaz.

```
valorpH=maestro.request("ph");
valorORP=maestro.request("orp");
valorEC=maestro.request("ec");
valorTemp=maestro.request("temp");
NDataManager::m_navData.HDGD=maestro.request("comp");

Serial.print( F( "VALORPH:" ) );
Serial.print(valorpH);
Serial.println( ';' );
Serial.print( F( "VALORORP:" ) );
Serial.print(valorORP);
Serial.println( ';' );
Serial.print( F( "VALOREC:" ) );
Serial.print(valorEC);
Serial.println( ';' );
Serial.print( F( "VALORTEMP:" ) );
Serial.print(valorTemp);
Serial.println( ';' );
Serial.print( F( "COMPASSYAW:" ) );
Serial.print(NDataManager::m_navData.HDGD);
Serial.println( ';' );
```

Como se observa en el código, se modifica el valor de la variable global «HDGD» encargada de almacenar el valor de la brújula por el nuevo valor.

Para la implementación del sistema de iluminación se crea una nueva función que, cuando recibe el comando encargado de activar y desactivar las luces, envía una petición al esclavo.

```
  if( commandIn.Equals( "ledaux" ) )
{
  int value = commandIn.m_arguments[1];
  if (value==1){
    maestro.request("ledon");
  }else{
    maestro.request("ledoff");
  }
}
```

10.7. Carcasa exterior

Para la realización del proyecto, se ha realizado un prototipo de carcasa donde poder encapsular la placa Arduino, así como todos sus componentes. Al ser un robot acuático este diseño debe extremar el cuidado y realizar estudios tanto de flotabilidad como de estanquidad.

La estructura donde se encapsula el sistema consiste en una carcasa exterior que se acopla sobre el ROV y cuatro tubos, dos a cada lado. Los tubos inferiores están abiertos y su único cometido es sujetar y proteger los sensores de golpes.

Los tubos superiores se cierran herméticamente mediante tapones con juntas tóricas para que queden totalmente estancos. En cada uno de los 2 tapones delanteros se colocan cuatro LEDs que incrementan la iluminación externa. Están sellados mediante resina epoxi. Los tapones traseros sirven para pasar los cables entre los diferentes tubos y la conexión al módulo central. La estanquidad de estos cables ha sido reforzada forrándolos con cinta aislante autosoldable de caucho.

El tubo superior izquierdo contiene la placa Arduino NANO, la PCB y los circuitos de los sensores. En el tubo derecho se guarda la batería y el lector microSD, elementos a los que hay que acceder de forma más frecuente.

La distribución, longitud y diámetro de dichos tubos han sido calculados no solo para contener los elementos del sistema, sino para asegurar que afecta lo mínimo posible a la flotabilidad del ROV.

En la zona superior de la carcasa, sellados mediante resina epoxi, se colocan la brújula y el GPS. Se sitúan en la zona superior para que en el momento que el ROV esté cerca de la superficie, pueda adquirir la señal GPS y almacenarla.

Figura 10.19: ROV con la carcasa.

11. Planificación Temporal

11.1. Metodología de desarrollo

Se decide aplicar una metodología ágil. Este tipo de metodología se basa en la división del trabajo en determinadas secciones funcionales que se van revisando e incrementando. Su objetivo es detectar errores y corregirlos rápidamente, llegando a la solución del proyecto más rápido.

La metodología elegida es SCRUM. Esta metodología permite seguir un proceso incremental basado en la división y el desarrollo de funcionalidades que se validan y corrigen conforme se avanza en la realización del proyecto. Esta característica permite realizar pequeños cambios con flexibilidad.

En otras metodologías se debe realizar primero un prototipo funcional con el que realizar las pruebas, esto supone un mayor gasto de tiempo y coste que se ahorra al usar la metodología SCRUM.

Al abordar el problema en pequeños trabajos el desarrollo del mismo es más cuidadoso y menos propenso a cometer errores.

11.2. Planificación temporal del proyecto

Esta planificación abarca todo el desarrollo del proyecto. Desde la primera reunión para analizar la viabilidad del proyecto hasta la finalización del periodo de pruebas.

La planificación se divide en cinco grandes etapas.

11.2.1. Especificación del proyecto

La primera etapa del proyecto consiste en el estudio de viabilidad y especificación de requisitos. Tiene como objetivo definir unos objetivos finales.

11.2.2. Investigación y búsqueda de información

En esta etapa se realiza la investigación del sistema del robot y el estudio de las alternativas que surgen en la realización del proyecto.

Se estudia cada componente que se va a usar para el desarrollo del módulo externo y los sistemas de control, iluminación y comunicación.

11.2.3. Desarrollo físico

Esta etapa contempla todo el desarrollo de la electrónica y hardware. Se estudian las microcontroladoras que se usan y su conexión con cada componente. Además, se diseña y construye la placa PCB donde van a ir todos los elementos conectados.

Paralelamente, se desarrolla un prototipo de carcasa donde sujetar y almacenar el módulo externo y sus diversos componentes.

11.2.4. Desarrollo del software del sistema

En esta etapa se desarrolla todo el software necesario para que el sistema entero funcione. Abarca desde la modificación del software del robot para adaptarlo a las necesidades del proyecto hasta el desarrollo del nuevo software para el módulo externo.

Durante toda la etapa se realizan pruebas de los trabajos finalizados con el fin de arreglar errores.

11.2.5. Memoria del proyecto

La última etapa consiste en la realización de la memoria del proyecto. Esta etapa abarca casi toda la duración del proyecto, desde que empieza la investigación de los componentes hasta la finalización de las pruebas.

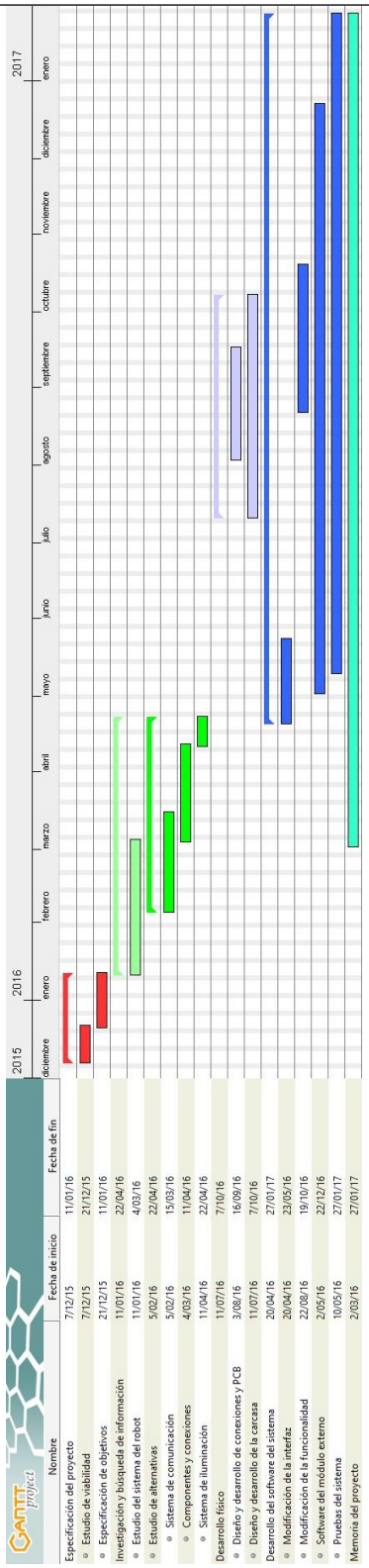


Figura 11.1: Diagrama de Gantt de la planificación temporal

12. Resumen del Presupuesto

El coste final del presupuesto para la realización del proyecto es la suma del presupuesto del ROV (1089 €) y el presupuesto del módulo externo (335,41 €), tal como se muestra en la tabla 12.1.

Descripción	**Precio**
Adquisición del ROV	1089 €
Realización del módulo externo	335,41 €
Total	1424,41 €

Cuadro 12.1: Presupuesto total

En el apartado 20 se desglosa el presupuesto detalladamente.

IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN HIDROLÓGICA EN UN ROBOT SUBACUÁTICO

REF: 000001

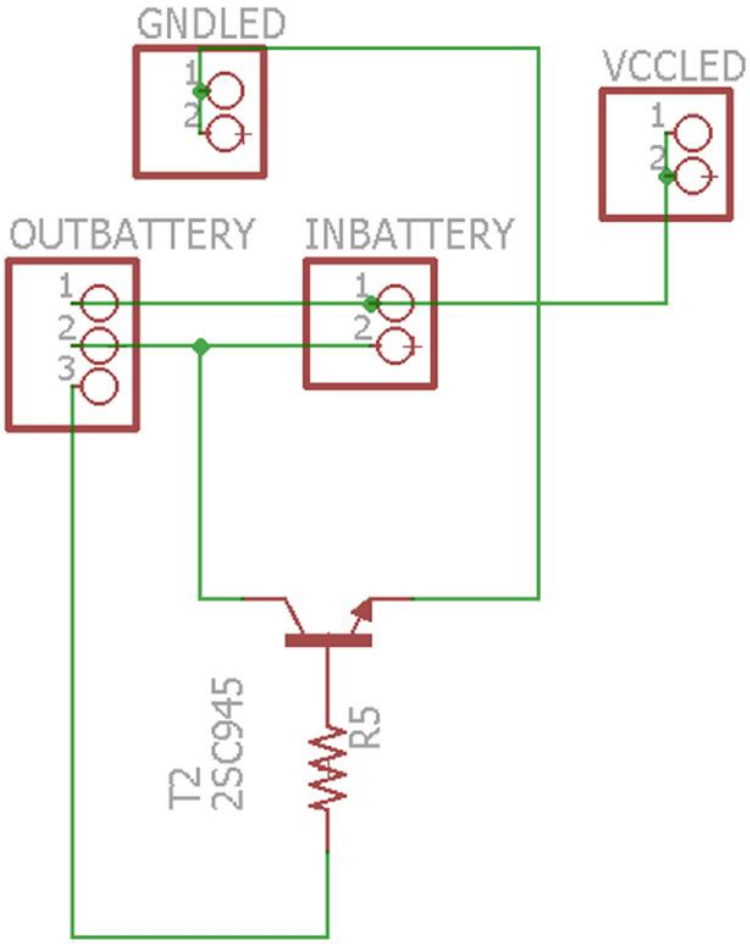
PLANOS

- **CLIENTE:** UNIVERSIDAD DE CÁDIZ (ESCUELA SUPERIOR DE INGENIERÍA)
  AVENIDA DE LA UNIVERSIDAD DE CÁDIZ Nº 10, 11519 PUERTO REAL
  956 48 32 00
  DIRECCION.ESI@UCA.ES

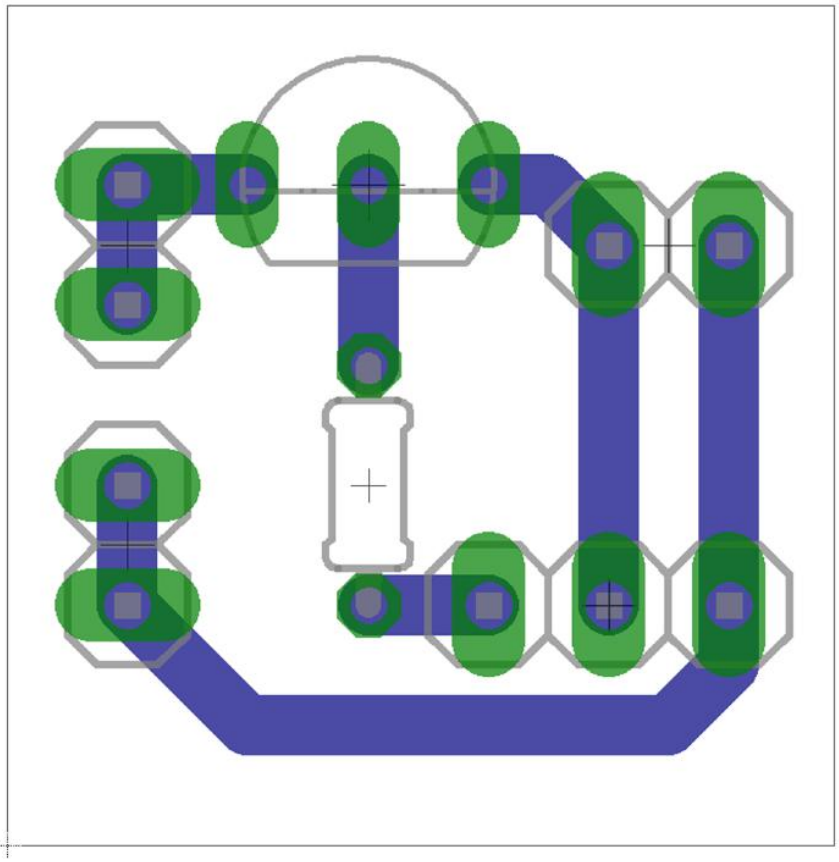
- **AUTOR:** ALEJANDRO CHACON PEREGRINO
  INGENIERO INFORMÁTICO
  DNI 32079315L
  ALEJANDRO.CHACONPEREGRINO@ALUM.UCA.ES

Cádiz, 2 de febrero de 2017

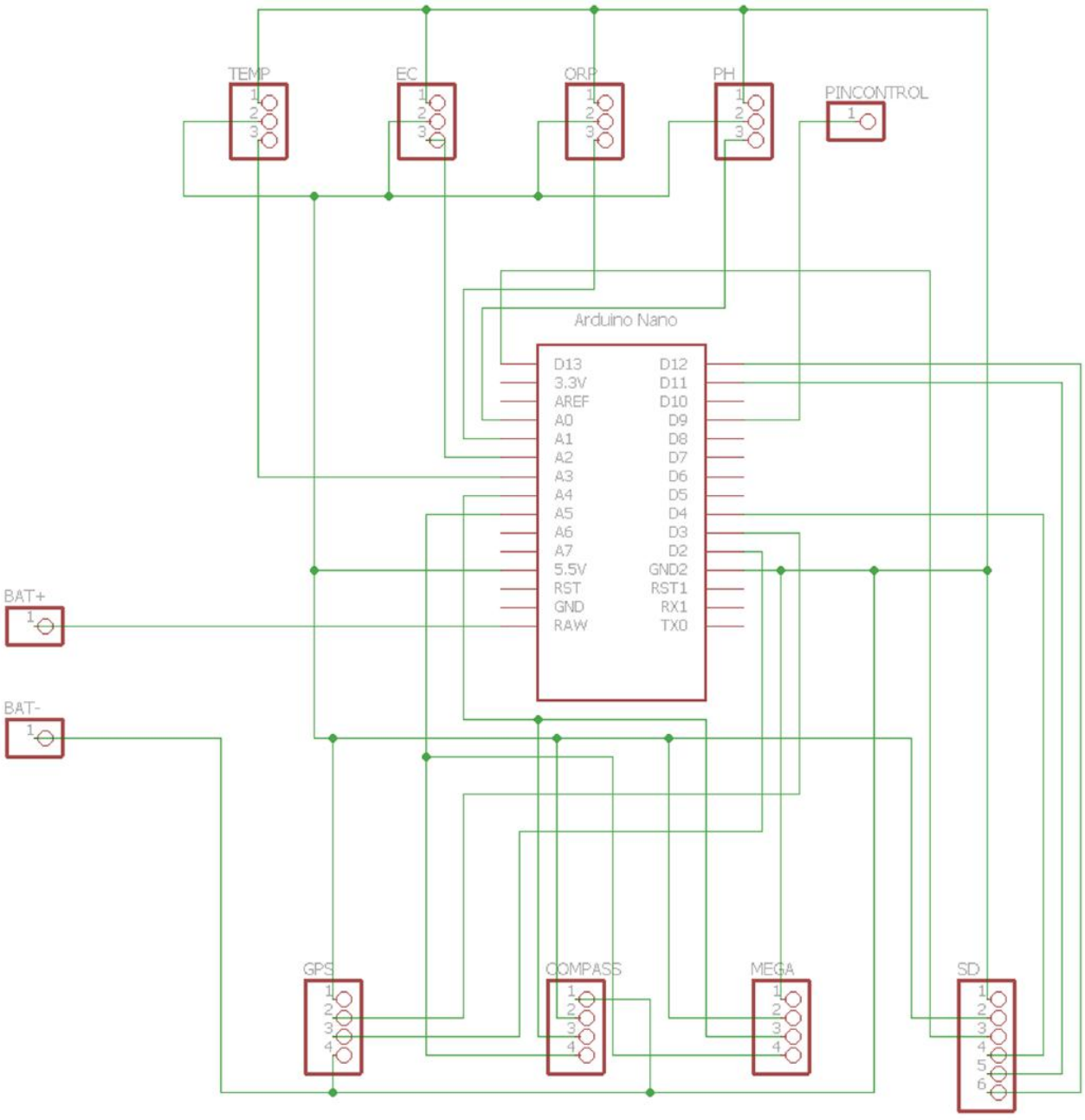
Índice



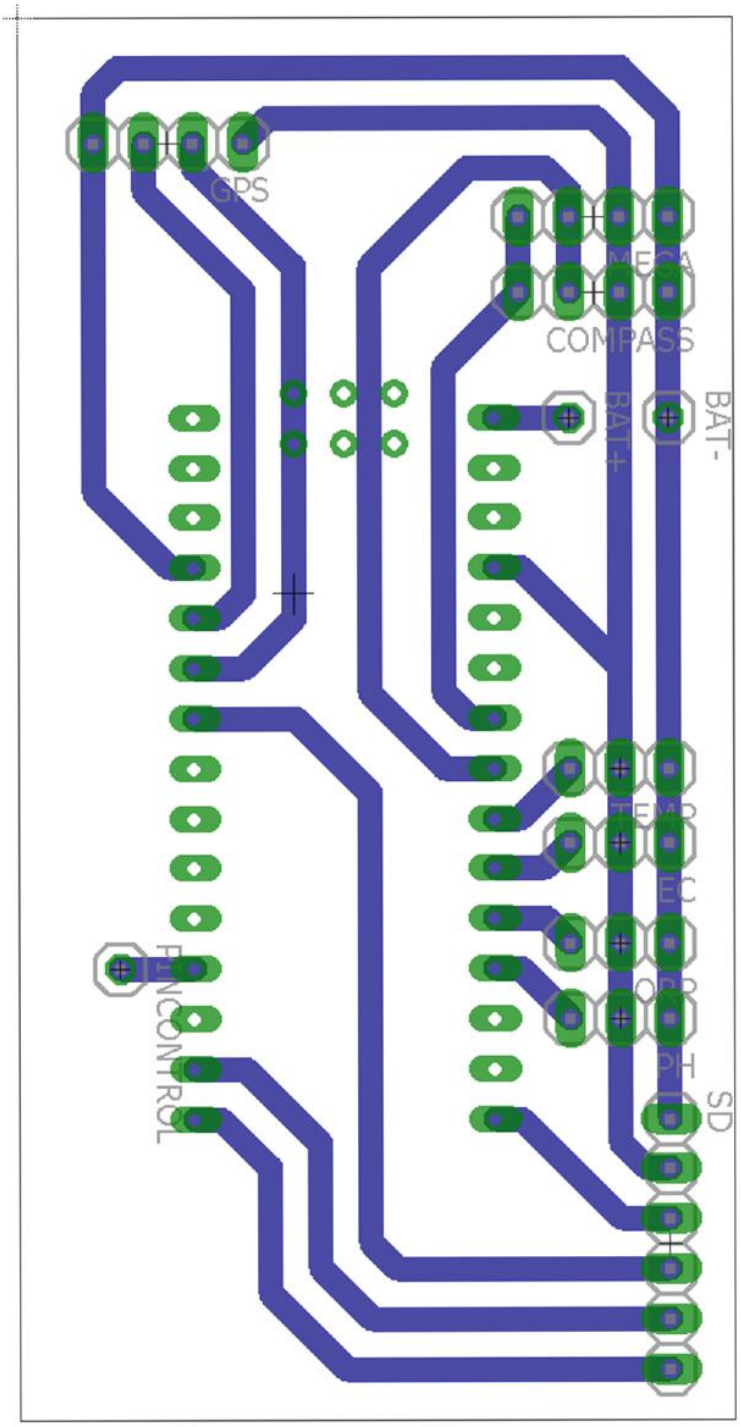
IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN HIDROLÓGICA EN UN ROBOT SUBACUÁTICO	
UNIVERSIDAD DE CÁDIZ - ESCUELA SUPERIOR DE INGENIERÍA	
TÍTULO: Esquemático PCB sistema de iluminación	REV: 1
Fecha: 10/01/17	Hoja: 1/1



IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN HIDROLÓGICA EN UN ROBOT SUBACUÁTICO	
UNIVERSIDAD DE CÁDIZ - ESCUELA SUPERIOR DE INGENIERÍA	
TÍTULO: Layout PCB sistema de iluminación	REV: 1
Fecha: 10/01/17	Hoja: 1/1



IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN HIDROLÓGICA EN UN ROBOT SUBACUÁTICO	
UNIVERSIDAD DE CÁDIZ - ESCUELA SUPERIOR DE INGENIERÍA	
TÍTULO: Esquemático PCB central	REV: 1
Fecha: 10/01/17	Hoja: 1/1



IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN HIDROLÓGICA EN UN ROBOT SUBACUÁTICO	
UNIVERSIDAD DE CÁDIZ - ESCUELA SUPERIOR DE INGENIERÍA	
TÍTULO: Layout PCB central	REV: 1
Fecha: 10/01/17	Hoja: 1/1

IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN HIDROLÓGICA EN UN ROBOT SUBACUÁTICO

REF: 000001

ANEXOS

- **CLIENTE:** UNIVERSIDAD DE CÁDIZ (ESCUELA SUPERIOR DE INGENIERÍA)
  AVENIDA DE LA UNIVERSIDAD DE CÁDIZ Nº 10, 11519 PUERTO REAL
  956 48 32 00
  DIRECCION.ESI@UCA.ES

- **AUTOR:** ALEJANDRO CHACON PEREGRINO
  INGENIERO INFORMÁTICO
  DNI 32079315L
  ALEJANDRO.CHACONPEREGRINO@ALUM.UCA.ES

Cádiz, 2 de febrero de 2017

Índice

17. Manual de usuario

En este manual se recopila toda la información relativa al uso y configuración del sistema.

17.1. Configuración inicial

Este paso sólo tendrá lugar la primera vez que conecta el ROV a su ordenador, aunque tendrá que repetirlo si cambia de equipo o reinstala el sistema operativo.

El ROV se conecta al PC mediante un cable Ethernet y un cable mini USB. Para ponerlo en marcha, se debe configurar la dirección IP de la tarjeta de red tal y como se indicará en el presente manual.

En Windows se encuentra en la siguiente ruta «Panel de control», «Redes e Internet», «Centro de redes y recursos compartidos», «Cambiar configuración del adaptador». Una vez en ese menú aparecen todas las interfaces de red disponibles, se elige «Ethernet» con el botón derecho y se hace click en «Propiedades».

Dentro de las propiedades, se selecciona «Protocolo de Internet versión 4 (TCP/IPv4)» y «Propiedades». Dentro del menú se debe configurar la IP como aparece en la figura 17.1.

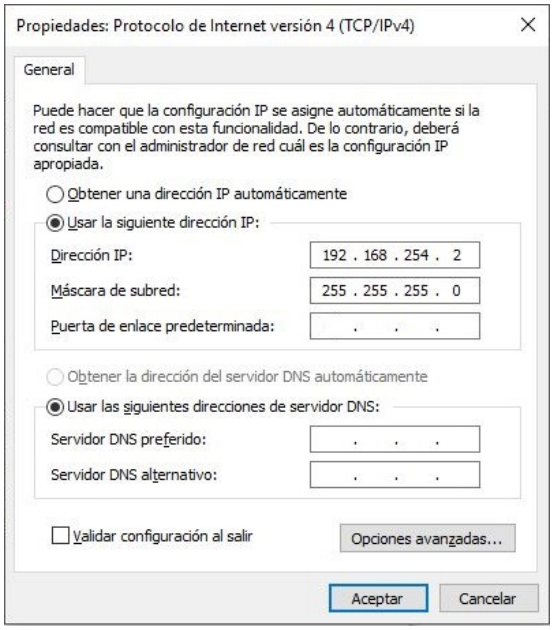


Figura 17.1: Configuración de IP.

17.1.1. Iniciar conexión

Una vez se ha configurado la dirección IP, se accede a la dirección «192.168.254.1» mediante el navegador Google Chrome (en otros navegadores puede dar fallos o no funcionar).

Aparecerá el «Dashboard», menú principal del ROV. Dentro de él se encuentran opciones de desarrollo y «OpenROV Cockpit». Esta última opción abre la interfaz lista para usar el ROV.

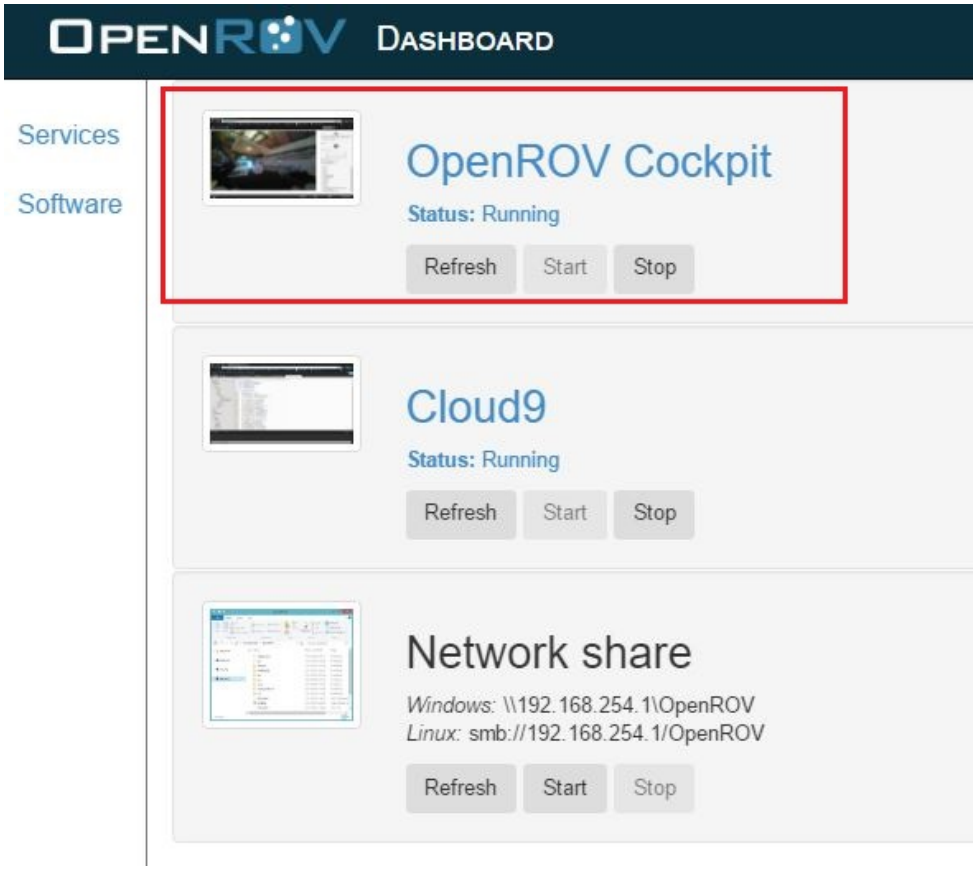


Figura 17.2: Menú principal del ROV.

17.2. Descripción de la interfaz

Una vez abierta la interfaz se observa una pantalla como la figura 17.3. Las diferentes zonas marcadas son detalladas a continuación.

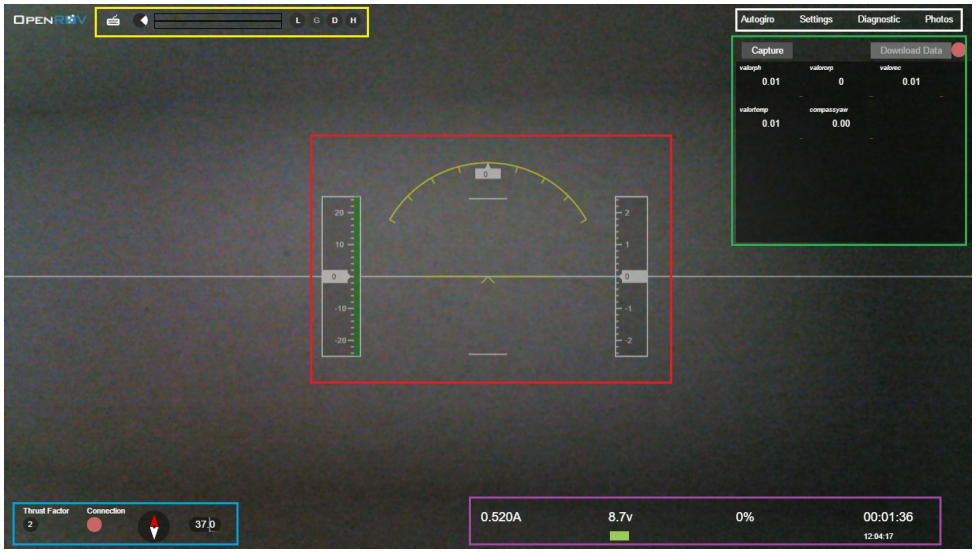


Figura 17.3: Interfaz del ROV.

17.2.1. Elementos de la interfaz

En la zona amarilla (figura 17.4) se encuentra un símbolo de teclado, al pulsarlo se muestran una serie de controles que serán descritos más abajo. También se encuentra el indicador de luz, que muestra la intensidad de las luces del ROV mediante una barra.

A su derecha se encuentran cuatro indicadores más que muestran si algunas funciones están actualmente activadas o desactivadas:

- L: Láser
- G: Se encuentra un mando conectado (gamepad).
- D: Control automático de giro.
- H: Control automático de profundidad.

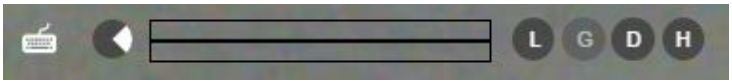


Figura 17.4: Zona amarilla.

La zona blanca muestra los menús de configuración, detallados en el apartado 17.2.2.

En la zona azul (figura 17.6) se encuentra la información relativa a los motores, brújula y conexión. De izquierda a derecha podemos encontrar:

- Thrust Factor: Muestra la potencia actual de los motores, siendo 1 la mínima y 5 la máxima. Esta opción se puede configurar como se indica en el apartado 17.2.3.
- Connection: Muestra la conexión a internet (actualmente en desarrollo por OpenROV).
- Compass: Muestra la brújula y su orientación.
- Latency: Muestra la latencia en la conexión con el PC. Mientras más alto sea el valor mayor retardo hay en la ejecución de órdenes.

Figura 17.5: Zona azul.

En la zona morada se encuentra la información relativa a la batería. Muestra el amperaje actual que está consumiendo el ROV y el porcentaje de carga de la batería. En el extremo derecho se muestra el tiempo que lleva el ROV en funcionamiento, junto a la hora actual.

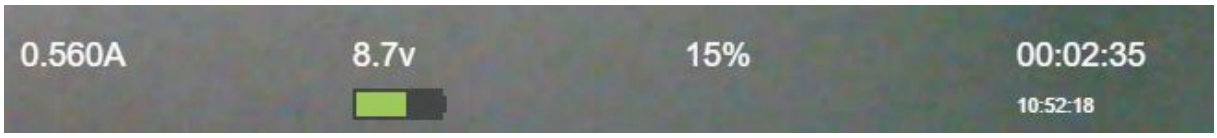


Figura 17.6: Zona morada.

En la zona roja (figura 17.7) se encuentra la telemetría. Mediante los gráficos se observan el nivel de profundidad al que está el ROV y la orientación en grados.

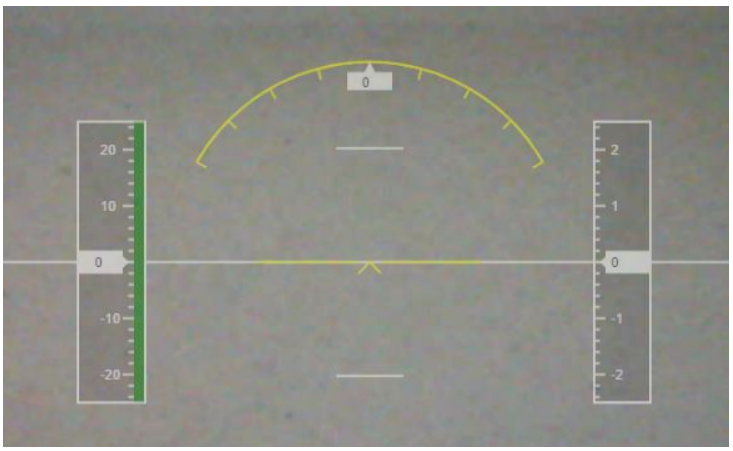


Figura 17.7: Zona roja.

A la derecha (zona verde, figura 17.8), aparecen una serie de gráficas que muestran la información captada por los sensores de la plataforma. Este menú está oculto hasta que el ratón pasa (se desliza) por encima. También dispone de un botón «Capture» para hacer capturas de pantalla sin los elementos de la interfaz.

Figura 17.8: Zona derecha.

17.2.2. Opciones disponibles en la interfaz

En la zona superior derecha se encuentran las cuatro opciones disponibles del menú que se detallan a continuación:

17.2.2.1. Diagnostics

Mediante este menú es posible configurar los tres motores invirtiendo su comportamiento, es decir, cambiar la dirección en la que se mueve un motor al pulsar la tecla asociada a él.

El motor «Port» es el derecho, «Starboard» es el izquierdo y «Vertical» es el superior.

Para esto, se puede activar la casilla «Reverse thruster» del motor correspondiente.

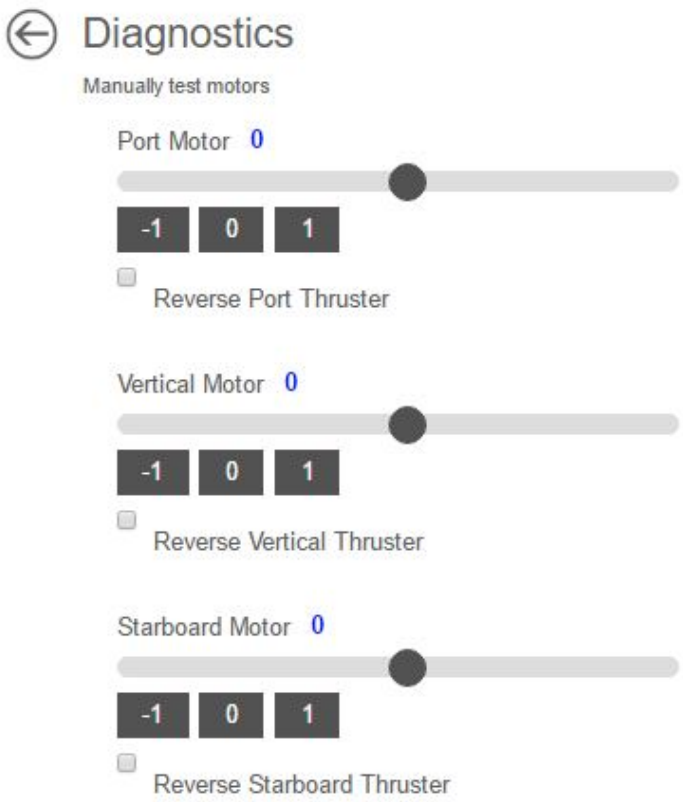


Figura 17.9: Menú Diagnostics.

17.2.2.2. Settings

Menú por defecto de la plataforma OpenROV, dispone de las opciones de configuración básicas para el sistema.

- Input Configuration: Muestra los botones predeterminados de control.
- Add new plugins: Permite buscar en internet por plugins de la comunidad.
- Plugins to enable/disable: Permite activar y desactivar los diferentes plugins instalados en el sistema.
- Software updates: Permite activar y desactivar el aviso de alertas de nuevas actualizaciones.
- Upload firmware: Permite recompilar los archivos Arduino del sistema. Necesario si se desea modificar el sistema.
- Battery Configuration: Permite elegir entre dos tipos diferentes de baterías precalibradas para mostrar su información en la interfaz.
- Google talk registration: Permite introducir los datos de la cuenta de Google del usuario e interactuar con el foro (actualmente en desarrollo).

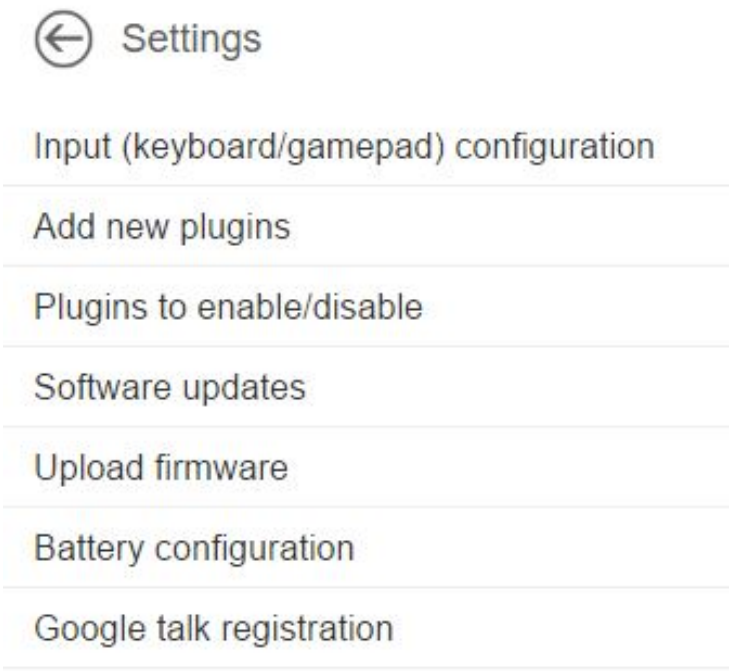


Figura 17.10: Menú Settings.

17.2.2.3. Autogiro

Mediante esta opción es posible configurar en qué grado de orientación se debe mantener el ROV durante el pilotaje.

Esta opción abre una nueva ventana con una rueda y una casilla en blanco. Permite elegir al usuario una orientación predeterminada o introducir los grados exactos. Una vez fijada la orientación se pulsa el botón «Confirmar» el valor se envía al sistema y podrá ser activado el autogiro. Si se activa sin elegir ninguna opción el valor por defecto es «0».

Figura 17.11: Menú Autogiro.

17.2.2.4. Photos

En este menú se encuentran almacenadas las fotos que se han tomado durante el pilotaje. Cuando se accede aparecen todas las fotos guardadas en el ROV y permite su almacenaje en el PC.

17.2.3. Control

Para el manejo de la mayoría de opciones y movimientos del ROV se utilizan las teclas del teclado. A continuación se muestran las teclas con su funcionamiento:

- Control de movimiento del ROV:
  - Flecha superior: Movimiento hacia delante.
  - Flecha inferior: Movimiento hacia atrás.
  - Flecha derecha: Gira hacia la derecha.
  - Flecha izquierda: Gira hacia la izquierda.
  - Shift: Activa el motor superior para bajar.
  - Control: Activa el motor superior para subir.
- Cámara:
  - Q: Movimiento de la cámara hacia arriba.
  - Z: Movimiento de la cámara hacia abajo.
  - A: La cámara se centra.
  - C: Hace una captura de pantalla sin interfaz.
- Luces y láseres:
  - L: Activa/Desactiva láseres.
  - I: Activa/Desactiva totalmente las luces internas.
  - O: Disminuye un nivel de intensidad las luces internas.
  - P: Aumenta un nivel de intensidad las luces internas.
  - 9: Activa/Desactiva las luces del módulo externo.
- Movimientos automáticos:
  - 7: Activa/Desactiva el movimiento recto.
  - 6: Activa/Desactiva el control de giro (permanece sin girar).
  - H: Activa/Desactiva el control de profundidad (permanece sin bajar ni subir).
- Potencia del motor:
  - 1-5: Modifica la potencia a la que funciona el motor, siendo 1 la más baja y 5 la más alta.

17.2.4. Extracción de batería y tarjeta SD

La batería y la tarjeta se encuentran en el tubo derecho del ROV. Para su correcta extracción se debe desenroscar el tapón frontal, donde se encuentran los LEDs.

La batería se debe desenchufar de su conector para cargarla. De otra forma, el sistema podría sufrir daños causados por la tensión.

La tarjeta microSD se encuentra conectada al lector, para su extracción es necesario apretarla suavemente hacia dentro hasta escuchar un «click». Al soltarla, saldrá automáticamente.

18. Librerías y códigos relevantes del sistema

18.1. Librerías

18.1.1. Sensor pH

18.1.1.1. sensorpH.h

```
class sensorpH{
  public:
    sensorpH(byte pinph);
    float getpH();
  private:
    byte sensorPin;
};
```

18.1.1.2. sensorph.cpp

```
#include <Arduino.h>
#include "sensorpH.h"

#define OFFSET 0  //Numero de calibracion.

int buf[10];    //Vector que almacena 10 medidas para realizar su media.

//Constructor de la clase que recibe el pin.
sensorpH::sensorpH(byte pinph){
  sensorPin = pinph;
}

//Funcion que calcula el pH.
float sensorpH::getpH(){
  int temp;   //Variable temporal para el calculo de la media.
  //Se recopilan 10 datos y se guardan en un vector.
  for(int i=0;i<10;i++){
    buf[i]=analogRead(sensorPin);
  }
  //Se ordena el vector de menor a mayor.
  for(int i=0;i<10;i++){
    for(int j=0;j<10;j++){
      if(buf[i]>buf[j]){
        temp=buf[i];
        buf[i]=buf[j];
        buf[j]=temp;
      }
    }
  }
```

```
30    unsigned long int avgValue=0;    //Variable que almacena el valor medio de las
        medidas.
31    //Se escogen las 6 medidas centrales y se hace la media.
32    for(int i=2;i<8;i++){
33      avgValue+=buf[i];
34    }
35    avgValue=avgValue/6;
36    //Se convierte el valor en milivoltios y luego en el valor pH.
37    float phValue=(float)avgValue*5.0/1024;
38    phValue=3.5*phValue+OFFSET;
39    return phValue;
40  }
```

18.1.2. Sensor ORP

18.1.2.1. sensorORP.h

```
1  class sensorORP{
2    public:
3      sensorORP(byte pin);
4      float getORP();
5    private:
6      double avergearray(int* arr, int number);
7      byte orpPin;
8  };
```

18.1.2.2. sensorORP.cpp

```
1  #include <Arduino.h>
2  #include "sensorORP.h"
3
4  #define VOLTAGE 5.00    //Voltaje del sistema.
5  #define ArrayLenth  40    //Numero de lecturas.
6  #define OFFSET 0  //Numero de calibracion.
7
8  double orpValue;  //Valor final de ORP.
9  int orpArray[ArrayLenth]; //Vector donde se guardan las lecturas.
10
11 //Constructor de la clase que recibe el pin.
12 sensorORP::sensorORP(byte pin){
13   orpPin=pin;
14 }
15
16 //Funcion que calcula el ORP.
17 float sensorORP::getORP(){
18   int orpArrayIndex=0;
19   //Se recopilan las lecturas en un vector.
20   for(int i=0;i<ArrayLenth;i++){
21     orpArray[orpArrayIndex++]=analogRead(orpPin);
22   }
23   //Se realiza la media y se guarda el valor.
24   orpValue=((30*(double)VOLTAGE*1000)-(75*avergearray(orpArray, ArrayLenth)*
       VOLTAGE*1000/1024))/75-OFFSET;
25   return orpValue;
26 }
27
```

```
28 //Funcion que ordena el vector y calcula la media.
29 double sensorORP::avergearray(int* arr, int number){
30   int i;
31   int max,min;
32   double avg;
33   long amount=0;
34   if(arr[0]<arr[1]){
35     min = arr[0];max=arr[1];
36   }
37   else{
38     min=arr[1];max=arr[0];
39   }
40   for(i=2;i<number;i++){
41     if(arr[i]<min){
42       amount+=min;
43       min=arr[i];
44     }else {
45       if(arr[i]>max){
46         amount+=max;
47         max=arr[i];
48       }else{
49         amount+=arr[i];
50       }
51     }
52   }
53   avg = (double)amount/(number-2);
54   return avg;
55 }
```

18.1.3. Sensor EC

18.1.3.1. sensorEC.h

```
1 class sensorEC{
2   public:
3     sensorEC(byte pinec);
4     void getVal();
5     float getEC(){return ecVal;}
6     float getTemp(){return tempVal;}
7   private:
8     byte ECsensorPin;
9     float ecVal;
10    float tempVal;
11    float TempProcess(bool ch);
12 };
```

18.1.3.2. sensorEC.cpp

```
1 #include <Arduino.h>
2 #include "sensorEC.h"
3 #include <OneWire.h>
4
5 #define StartConvert 0
6 #define ReadTemperature 1
7
8 const byte numReadings = 20;      //Numero de lecturas realizadas.
```

```
byte DS18B20_Pin = A3;  //Pin del sensor de temperatura.
unsigned int readings[numReadings]; //Vector donde se guardan los valores.
unsigned long AnalogValueTotal = 0; //Valor de las lecturas.
unsigned int AnalogAverage = 0,averageVoltage=0;  //Valor medio de las lecturas y voltaje.
float temperature,ECcurrent;  //Temperatura y EC final.

//Iniciacion del sensor de temperatura.
OneWire ds(DS18B20_Pin);

//Constructor de la clase que recibe el pin.
sensorEC::sensorEC(byte pinec){
  ECsensorPin = pinec;
}

//Funcion que calcula el EC y Temperatura.
void sensorEC::getVal(){
  byte index = 0;
  //Se realiza la lectura de datos, se guardan en el vector y se realiza la media.
  for(int i=0;i<numReadings;i++){
    AnalogValueTotal = AnalogValueTotal - readings[index];
    readings[index] = analogRead(ECsensorPin);
    AnalogValueTotal = AnalogValueTotal + readings[index];
    index = index + 1;
    AnalogAverage = AnalogValueTotal / numReadings;
  }

  //Se lee la temperatura y se resetea la memoria para la siguiente lectura de temperatura.
  temperature = TempProcess(ReadTemperature);
  TempProcess(StartConvert);

  //Se realiza la conversion a EC mediante las formulas de compensacion facilitadas por el fabricante.
  averageVoltage=AnalogAverage*(float)5000/1024;
  float TempCoefficient=1.0+0.0185*(temperature-25.0);
  float CoefficientVolatge=(float)averageVoltage/TempCoefficient;
  if(CoefficientVolatge<=448)ECcurrent=6.84*CoefficientVolatge-64.32;
  else if(CoefficientVolatge<=1457)ECcurrent=6.98*CoefficientVolatge-127;
  else ECcurrent=5.3*CoefficientVolatge+2278;
  ECcurrent/=1000;
  ecVal=(float)ECcurrent;
  tempVal=(float)temperature;
}

//Funcion facilitada por el fabricante que calcula la temperatura.
float sensorEC::TempProcess(bool ch) {
  static byte data[12];
  static byte addr[8];
  static float TemperatureSum;
  if(!ch){
    ds.reset();
    ds.select(addr);
    ds.write(0x44,1);
  }
  else{
    byte present = ds.reset();
```

```
63      ds.select(addr);
64      ds.write(0xBE);
65      for (int i = 0; i < 9; i++) {
66        data[i] = ds.read();
67      }
68      ds.reset_search();
69      byte MSB = data[1];
70      byte LSB = data[0];
71      float tempRead = ((MSB << 8) | LSB);
72      TemperatureSum = tempRead / 16;
73    }
74    return TemperatureSum;
75    }
```

18.1.4. Comunicación I2C

18.1.4.1. comi2c.h

```
1 #include <Arduino.h>
2
3 class comi2c{
4   public:
5     comi2c(byte id);
6     float request(String op);
7   private:
8     int receiveResponse();
9     byte addr;
10 };
```

18.1.4.2. comi2c.cpp

```
1 #include "comi2c.h"
2 #include <Wire.h>
3
4 //Constructor que recibe la direccion del esclavo.
5 comi2c::comi2c(byte id){
6   addr=id;  //Se asigna el identificador del esclavo para su posterior uso.
7 }
8
9 //Funcion que desempaqueta la respuesta para transformarla en un entero.
10 int comi2c::receiveResponse(){
11   Wire.beginTransmission(addr); //Se inicia la transmision con el esclavo.
12   int receivedValue = Wire.read() << 8 | Wire.read();   //Desempaqueta la
       respuesta.
13   Wire.endTransmission(true);   //Finaliza la transmision.
14   return receivedValue;
15 }
16
17 //Funcion que envia la orden al esclavo.
18 float comi2c::request(String op){
19   int type=0; //Existen dos tipos dependiendo del tipo de dato. 1 = int, 0 =
       float.
20   byte DataPacket[1];
21   if(op=="ph"){
22     DataPacket[0] = 0;
23     type=1;
```

```
24    }
25    if(op=="orp"){
26      DataPacket[0] = 1;
27      type=0;
28    }
29    if(op=="ec"){
30      DataPacket[0] = 2;
31      type=1;
32    }
33    if(op=="temp"){
34      DataPacket[0] = 3;
35      type=1;
36    }
37    if(op=="comp"){
38      DataPacket[0] = 4;
39      type=0;
40    }
41    if(op=="ledon"){
42      DataPacket[0] = 10;
43      type=0;
44    }
45    if(op=="ledoff"){
46      DataPacket[0] = 11;
47      type=0;
48    }
49    Wire.beginTransmission(addr); //Inicia la transmision para enviar el comando.
50    Wire.write(DataPacket, 1);    //Envia el comando.
51    Wire.endTransmission(true);   //Finaliza la transmision.
52    delay(10);
53    float response = receiveResponse();   //Cuando la respuesta es desempaquetada
        se recibe.
54    //Dependiendo del tipo de dato se realiza la siguiente operacion.
55    if(type==0){
56      return response;
57    }else return response/100;
58  }
```

18.2. Códigos del sistema

18.2.1. Maestro

```
#include "AConfig.h"
#if(HAS_STD_PRUEBA)

#include "comi2c.h"
#include <Arduino.h>
#include "CPrueba.h"
#include "CPin.h"
#include "NConfigManager.h"
#include "NModuleManager.h"
#include "NCommManager.h"
#include "NDataManager.h"
#if(HAS_STD_CAPE)
#include "CCape.h"
#endif
#if(HAS_OROV_CONTROLLERBOARD_25)
#include "CControllerBoard.h"
#endif

namespace {
  //Variables para almacenar los valores.
  float valorpH, valorEC, valorTemp;
  int valorORP;
  //Inicializacion de la clase.
  comi2c maestro(8);
}

//Funcion que inicializa la configuracion inicial.
void CPrueba::Initialize() {
  NConfigManager::m_capabilityBitmask |= ( 1 << PRUEBA_CAPABLE );
}

//Funcion que se ejecuta continuamente.
void CPrueba::Update(CCommand& commandIn) {
  //Se manda la orden para recibir el valor de los sensores.
  valorpH=maestro.request("ph");
  valorORP=maestro.request("orp");
  valorEC=maestro.request("ec");
  valorTemp=maestro.request("temp");
  NDataManager::m_navData.HDGD=maestro.request("comp"); //Se modifica el valor
    de la brĂşjula por defecto por el nuevo.
  //Se envian los valores para mostrarlos en la interfaz.
  Serial.print( F("VALORPH:"));
  Serial.print(valorpH);
  Serial.println(';');
  Serial.print( F("VALORORP:"));
  Serial.print(valorORP);
  Serial.println(';');
  Serial.print(F("VALOREC:"));
  Serial.print(valorEC);
  Serial.println(';');
  Serial.print(F("VALORTEMP:"));
  Serial.print(valorTemp);
  Serial.println(';');
```

```
53    Serial.print(F("COMPASSYAW:"));
54    Serial.print(NDataManager::m_navData.HDGD);
55    Serial.println(';');
56    //Si se recibe el comando se envia la orden de encender o apagar el sistema
        de iluminacion.
57    if(commandIn.Equals("ledaux")) {
58      int value = commandIn.m_arguments[1];
59      if (value==1) maestro.request("ledon");
60      else maestro.request("ledoff");
61    }
62  }
63  #endif
```

18.2.2. Esclavo

```
1  #include <Wire.h> //Libreria I2C.
2  #include <SD.h> //Libreria para manejar la SD.
3  //Librerias de sensores.
4  #include <sensorORP.h>
5  #include <sensorpH.h>
6  #include <sensorEC.h>
7  #include <HMC5883L_Simple.h>
8
9  //Librerias para el GPS.
10 #include <TinyGPS++.h>
11 #include <SoftwareSerial.h>
12
13 const byte SlaveDeviceId = 8; //ID del esclavo para la conexion I2C.
14 byte LastMasterCommand = 0; //Variable donde se guarda el comando enviado por
       el maestro.
15 const byte pinSD = 4;  //Variable con el pin al que se conecta la SD.
16 //Contador utilizado para el almacenamiento en la SD.
17 long intervalo = 3000;
18 long tiempo = 0;
19 long tiempoAnterior = 0;;
20
21 //Constructores de los objetos sensores pasandole los pines al que estan
       conectados.
22 sensorORP sensor_orp(A2);
23 sensorEC sensor_ec(A1);
24 sensorpH sensor_ph(A0);
25 HMC5883L_Simple Compass;
26 TinyGPSPlus gps;
27 SoftwareSerial ss(3, 2);
28
29 //Inicializacion de variables
30 int valorORP, valorEC, valorCompass;  //Variables para los sensores.
31 float valorpH, valorTemp;
32 float valorLongitud, valorLatitud; //Variables para el GPS.
33 File myFile;  //Variable para el archivo de la SD.
34
35 void setup(){
36   //Inicializacion de los componentes.
37   sensor_orp.setup();
38   sensor_ec.setup();
39   sensor_ph.setup();
40
```

```
41    //Opciones para calibrar el compass y muestre el verdadero norte.
42    Compass.SetDeclination(23, 35, 'E');
43    Compass.SetSamplingMode(COMPASS_SINGLE);
44    Compass.SetScale(COMPASS_SCALE_130);
45    Compass.SetOrientation(COMPASS_HORIZONTAL_X_NORTH);
46    ss.begin(4800);       //Inicio del GPS.
47    SD.begin(pinSD);      //Inicio de la SD.
48    pinMode(9, OUTPUT); //Pin para el sistema de iluminacion configurado como
        salida.
49
50    Wire.begin(SlaveDeviceId);  //Se inicia la conexion I2C con su ID.
51    Wire.onReceive(receiveDataPacket); //Funcion para recibir peticiones I2C.
52    Wire.onRequest(slavesRespond);  //Funcion para responder por I2C.
53  }
54
55  void loop(){
56    delay(100); //Delay de seguridad.
57    //Funciones de las librerias de sensores para obtener los valores y tenerlos
        listos en caso de que el maestro los pida.
58    valorORP = sensor_orp.getORP();
59    valorpH = sensor_ph.getpH();
60    sensor_ec.getVal();
61    valorEC = sensor_ec.getEC();
62    valorTemp = sensor_ec.getTemp();
63    valorCompass=Compass.GetHeadingDegrees();
64
65    //Cada 30 segundos se guarda la informaciÃłn en la SD.
66    tiempo = millis();
67      if(tiempo - tiempoAnterior > intervalo) {
68        tiempoAnterior = tiempo;
69        //Si la comunicaciÃłn con el GPS esta disponible se reciben los datos y
        se guardan.
70        if (ss.available() > 0){
71          valorLongitud = gps.location.lng();
72          valorLatitud = gps.location.lat();
73        }
74
75        //Se abre el archivo y se guarda la informacion.
76        myFile = SD.open("info.txt", FILE_WRITE);
77        if (myFile) {
78          myFile.print("Posicion: ");
79          myFile.print(valorLongitud);
80          myFile.print(", ");
81          myFile.print(valorLatitud);
82          myFile.print(" - pH: ");
83          myFile.print(valorpH);
84          myFile.print(" - ORP: ");
85          myFile.print(valorORP);
86          myFile.print("mV - EC: ");
87          myFile.print(valorEC);
88          myFile.print("ms - Temperatura: ");
89          myFile.print(valorTemp);
90          myFile.println(" ÂşC");
91        }
92        myFile.close();
93      }
94  }
95
```

```
96 //Funcion que recibe el comando de la accion a ejecutar del maestro.
97 void receiveDataPacket(int howMany){
98   LastMasterCommand = Wire.read();
99 }
100
101 //Funcion para responder al maestro con el dato solicitado.
102 void slavesRespond(){
103   int returnValue = 0; //Variable que se va a enviar al maestro como respuesta.
104
105   //Switch encargado de guardar en la variable el valor deseado para enviarla
        posteriormente.
106   //0 = pH
107   //1 = ORP
108   //2 = EC
109   //3 = Temperatura
110   //4 = Brujula
111   //Y de realizar las siguientes funciones.
112   //10 = Encender las luces
113   //11 = Apagar las luces
114   switch(LastMasterCommand){
115     case 0:
116       returnValue = valorpH*100;
117       break;
118     case 1:
119       returnValue = valorORP;
120       break;
121     case 2:
122       returnValue = valorEC*100;
123       break;
124     case 3:
125       returnValue = valorTemp*100;
126       break;
127     case 4:
128       returnValue = valorCompass;
129       break;
130     case 10:
131       digitalWrite(9, HIGH);
132       break;
133     case 11:
134       digitalWrite(9, LOW);
135       break;
136   }
137   //Se divide el dato en un buffer de dos bytes para poder enviar valores
        negativos y mayores a 255 a traves de I2C.
138   byte buffer[2];
139   buffer[0] = returnValue >> 8;
140   buffer[1] = returnValue & 255;
141   Wire.write(buffer, 2); //Se envia el buffer al maestro.
142 }
```

IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN HIDROLÓGICA EN UN ROBOT SUBACUÁTICO

REF: 000001

ESPECIFICACIONES DEL SISTEMA

- **CLIENTE:** UNIVERSIDAD DE CÁDIZ (ESCUELA SUPERIOR DE INGENIERÍA)
  AVENIDA DE LA UNIVERSIDAD DE CÁDIZ Nº 10, 11519 PUERTO REAL
  956 48 32 00
  DIRECCION.ESI@UCA.ES

- **AUTOR:** ALEJANDRO CHACON PEREGRINO
  INGENIERO INFORMÁTICO
  DNI 32079315L
  ALEJANDRO.CHACONPEREGRINO@ALUM.UCA.ES

Cádiz, 2 de febrero de 2017

Índice

19. Especificaciones del Sistema

19.1. Objetivos del sistema

OBJ-01	Monitorización de variables.
Descripción	El sistema debe solicitar al módulo externo las variables que leen los sensores de manera periódica y mostrarlas al usuario en la pantalla.
Importancia	Alta.
Estabilidad	Alta.
Comentarios	Ninguno.

Cuadro 19.1: Objetivo 1 del sistema: Monitorización de variables.

OBJ-02	Almacenamiento de datos.
Descripción	El módulo externo debe guardar todos los datos de los sensores de forma periódica.
Importancia	Alta.
Estabilidad	Alta.
Comentarios	Ninguno.

Cuadro 19.2: Objetivo 2 del sistema: Almacenamiento de datos.

OBJ-03	Iluminación externa.
Descripción	El sistema debe ofrecer las funcionalidades para controlar la iluminación externa: ▪ Encender iluminación externa. ▪ Apagar iluminación externa.
Importancia	Alta.
Estabilidad	Alta.
Comentarios	Ninguno.

Cuadro 19.3: Objetivo 3 del sistema: Iluminación externa.

OBJ-04	Posicionamiento del ROV.
Descripción	El sistema debe ofrecer las funcionalidades para posicionar el ROV: ▪ Guardar los grados solicitados por el usuario. ▪ Activar control automático. ▪ Desactivar control automático.
Importancia	Alta.
Estabilidad	Alta.
Comentarios	Ninguno.

Cuadro 19.4: Objetivo 4 del sistema: Posicionamiento del ROV.

19.2. Descripción de actores

ACT-01	Usuario.
Descripción	Usuario final del sistema.

Cuadro 19.5: Actor 01: usuario del sistema.

19.3. Requisitos funcionales

FRQ-01	Mostrar variables en pantalla.
Objetivos asociados	▪ OBJ-01 Monitorización de variables.
Descripción	El sistema debe mostrar las variables de los sensores en la pantalla de forma periódica.

Cuadro 19.6: Requisito Funcional 01: Mostrar variables en pantalla.

FRQ-02	Almacenar información.
Objetivos asociados	▪ OBJ-02 Almacenamiento de datos .
Descripción	El sistema debe almacenar la información en un medio externo de forma periódica.

Cuadro 19.7: Requisito Funcional 02: Almacenar información.

19.3.1. Diagrama de Casos de Uso

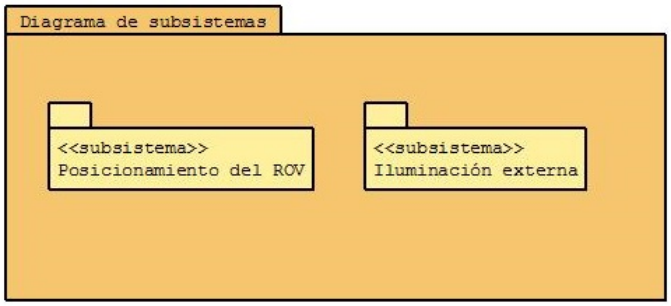


Figura 19.1: Diagrama de subsistemas.

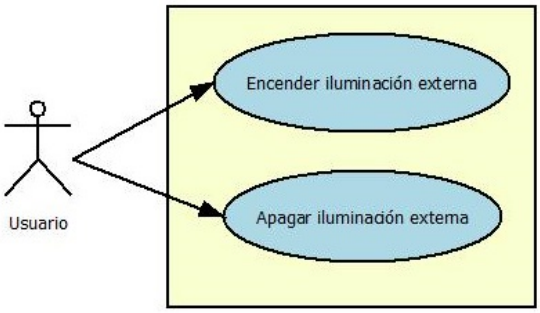


Figura 19.2: Diagrama de subsistema de iluminación.

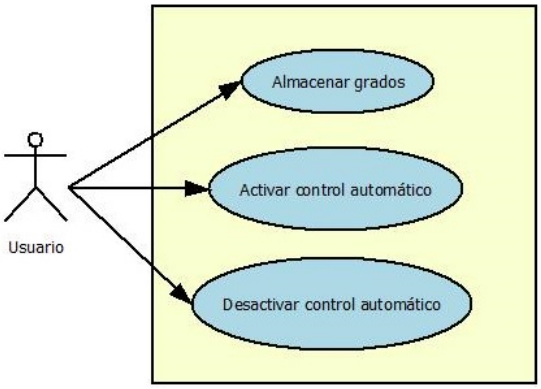


Figura 19.3: Diagrama de subsistema de poscionamiento.

19.3.2. Casos de Uso

UC-01	Encender iluminación externa.	
Objetivos asociados	▪ OBJ-03 Iluminación externa.	
Descripción	El sistema envía la señal de encendido a la iluminación externa.	
Precondición	El sistema de iluminación está inactivo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-01) pulsa el botón «encender luces».
	2	El sistema envía la señal para encender las luces al sistema de iluminación.
Postcondición	El sistema de iluminación permanece activo.	

Cuadro 19.8: Caso de Uso 13: Encender iluminación externa.

UC-02	Apagar iluminación externa.	
Objetivos asociados	▪ OBJ-03 Iluminación externa.	
Descripción	El sistema envía la señal de apagado a la iluminación externa.	
Precondición	El sistema de iluminación está activo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-01) pulsa el botón «apagar luces».
	2	El sistema envía la señal para apagar las luces al sistema de iluminación.
Postcondición	El sistema de iluminación permanece inactivo.	

Cuadro 19.9: Caso de Uso 02: Apagar iluminación externa.

UC-03	Almacenar grados.	
Objetivos asociados	▪ OBJ-04 Posicionamiento del ROV.	
Descripción	El sistema guarda los grados introducidos por el usuario.	
Precondición	Ninguna.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-01) pulsa el botón con los grados elegidos.
	2	El sistema guarda en una variable «grados» la elección.
Postcondición	Los grados elegidos por el usuario permanecen guardados en una variable.	

Cuadro 19.10: Caso de Uso 03: Almacenar grados.

UC-04	Activar control automático.	
Objetivos asociados	▪ OBJ-04 Posicionamiento del ROV.	
Descripción	El sistema debe mantener al ROV en una misma dirección.	
Precondición	El control automático está inactivo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-01) pulsa el botón «activar control automático»
	2	El sistema redirige automáticamente el ROV para permanecer en la trayectoria almacenada en la variable «grados».
Postcondición	El sistema mantiene constantemente al ROV en una trayectoria fija.	

Cuadro 19.11: Caso de Uso 04: Activar control automático.

UC-05	Desactivar control automático.	
Objetivos asociados	▪ OBJ-04 Posicionamiento del ROV.	
Descripción	El sistema deja girar el ROV al usuario.	
Precondición	El control automático está activo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-01) pulsa el botón «desactivar control automático»
	2	El sistema desbloquea los motores y permite al actor Usuario (ACT-01) girar.
Postcondición	El sistema permite modificar la trayectoria del ROV libremente.	

Cuadro 19.12: Caso de Uso 05: Desactivar control automático.

IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN HIDROLÓGICA EN UN ROBOT SUBACUÁTICO

REF: 000001

PRESUPUESTO

- **CLIENTE:** UNIVERSIDAD DE CÁDIZ (ESCUELA SUPERIOR DE INGENIERÍA)
  AVENIDA DE LA UNIVERSIDAD DE CÁDIZ Nº 10, 11519 PUERTO REAL
  956 48 32 00
  DIRECCION.ESI@UCA.ES

- **AUTOR:** ALEJANDRO CHACON PEREGRINO
  INGENIERO INFORMÁTICO
  DNI 32079315L
  ALEJANDRO.CHACONPEREGRINO@ALUM.UCA.ES

Cádiz, 2 de febrero de 2017

Índice

20. Presupuesto

20.1. Presupuesto para la adquisición del ROV

En la tabla 20.1 se incluye el presupuesto de la adquisición del vehículo de bajo coste OpenROV, IVA incluido.

Artículo	Cantidad	Precio total
Kit de montaje OpenROV [31]	1	1.089 €
Total		1089 €

Cuadro 20.1: Presupuesto del ROV.

20.2. Presupuesto para la realización del módulo externo

En la tabla 20.2 se incluye el presupuesto total para la realización del módulo externo. Cada artículo incluye su precio individual y total, IVA incluido.

Artículo	Cantidad	Precio unitario	Precio total
Sensor pH DFRobots [15]	1	52,78 €	52,78 €
Sensor ORP DFRobots [14]	1	82,63 €	82,63 €
Sensor EC y Temperatura DFRobots [13]	1	64.85 €	64.85 €
Sensor magnético GY-273 [19]	1	12,39 €	12,39 €
Lector microSD [18]	1	2,20 €	2,20 €
Módulo GPS GY-NEO6MV2 [16]	1	13,39 €	13,39 €
Batería LIPO 7,4V y 2800mAh Floureon [3]	1	29,99 €	29,99 €
Arduino NANO [37]	1	18,36 €	18,36 €
LED Blanco 5mm [39]	8	0,97 €	7,74 €
Kit 6x cable 1m AWG24 [17]	1	2,25 €	2,25 €
Resistencia de 33Ω [41]	4	0,18 €	0,73 €
Resistencia 100Ω [40]	1	0,19 €	0,19 €
Transistor NPN 2N2222 [21]	1	2,04 €	2,04 €
Balda rectangular de metacrilato [26]	1	19,95 €	19,95 €
Tubo rígido PVC evacuación 50mm y 2m [28]	1	3,40 €	3,40 €
Rosca PVC 50mm [27]	4	0,88 €	3,52 €
Cinta eléctrica autosoldable de Goma [2]	1	19 €	19 €
Total			335,41 €

Cuadro 20.2: Presupuesto del módulo externo.

20.3. Resumen del presupuesto

El coste final del presupuesto para la realización del proyecto es la suma del presupuesto del ROV (1089 €) y el presupuesto del módulo externo (335,41 €), tal como se muestra en la tabla 20.3.

Descripción	**Precio**
Adquisición del ROV	1089 €
Realización del módulo externo	335,41 €
Total	1424,41 €

Cuadro 20.3: Presupuesto total